(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 November 2001 (22.11.2001)

PCT

(10) International Publication Number
WO 01/88761 A2

(51) International Patent Classification[7]: G06F 17/30

(21) International Application Number: PCT/US01/15575

(22) International Filing Date: 15 May 2001 (15.05.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/571,053 15 May 2000 (15.05.2000) US

(71) Applicant *(for all designated States except US)*: **INNOVATIVE COMMUNICATIONS TECHNOLOGIES, INC.** [US/US]; 9021 Gaither Road, Gaithersburg, MD 20877 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **MCCONNELL, Danny, Edward** [US/US]; 7506 Box Elder Court, McLean, VI 22102 (US). **JACOBSON, Jeffrey, Richard** [US/US]; 7825 Aberdeen Road, Bethesda, MD 20814 (US).

(74) Agents: **DARBY, George, E.** et al.; P.O. Box 10107, Greenville, SC 29603 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

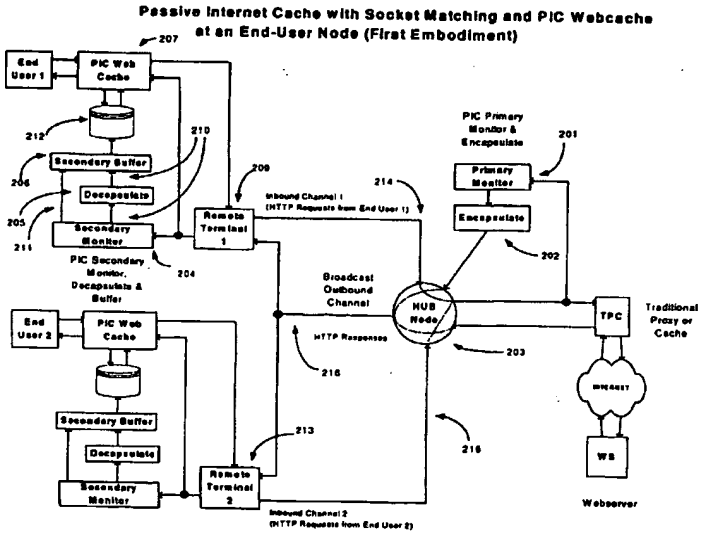
(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii)) for all designations*

[Continued on next page]

(54) Title: A SYSTEM AND METHOD FOR A PASSIVE INTERNET CACHE



(57) Abstract: The Passive Internet Cache ("PIC") system builds a global cache at or near end-user nodes, and substantially avoids response delays at end-user nodes arising from the exchange of request and response messages between a remote terminal and a hub node in a star topology network. Specifically, the PIC system "passively listens" using a secondary monitor (204) to all HTTP responses in an outbound channel (216) and enables end-user nodes interfaced with remote terminals (209) to use the contents of all HTTP responses in an outbound channel (216), reduces the repetitive transmission of the same HTTP response payloads over an outbound channel (216), and in dual channel networks also reduces the need to send HTTP requests over an inbound channel (214). The PIC system operates in any star topology network, including without limitation VSAT systems, Direct-to-Home satellite systems, and multichannel multipoint distribution service.

WO 01/88761 A2

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii)) for all designations*
— *of inventorship (Rule 4.17(iv)) for US only*

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

TITLE OF THE INVENTION

A SYSTEM AND METHOD FOR A PASSIVE INTERNET CACHE

BACKGROUND OF THE INVENTION

1. Field of the Invention

The Passive Internet Cache system pertains to data caches in "star topology", dual channel (send and receive), and single channel (receive only, or "broadcast"), networks that use the HyperText Transfer Protocol ("HTTP"). HTTP is very commonly used with Transmission Control Protocol/Internet Protocol ("TCP/IP") over the Internet or private networks that have a client/server architecture in which client software operated by an end user requests and receives information from one or more server computers. The structure of such requests and responses conforms with the specifications for HTTP. HTTP specifications are currently promulgated by the World Wide Web Consortium, Cambridge, Massachusetts.

An "HTTP request" from a World Wide Web browser, or other software application that uses HTTP messages, to a server computer contains a Uniform Resource Locator ("URL") to identify the requested information. The data contents at the address specified by a URL can be a complete Webpage, or components of a Webpage (text, graphics, sound, video, or other type of file). The "HTTP response" from the server to the browser contains the "contents" of the URL, but normally does not restate the URL. The cache that is included with commonly available Web browsers interoperates with the browser as follows: (a) an end-user clicks on a URL displayed in a browser or other window of the display of an electronic device, usually a computer, connected to the Internet; (b) the browser acts in conformity with the cache configuration settings of the browser, e.g., if the URL is stored in the built-in browser cache, the cache provides the contents of the stored URL to the browser unless required by the configuration settings to verify that the cached URL has not changed; if verification is required, or if the URL is not stored in browser cache, the cache sends an HTTP request message (usually including a "GET", or "GET IF-MODIFIED-SINCE" command) to request the URL from the host computer specified in the URL; (c) if the disk space used by browser cache exceeds a user-defined size, the cache deletes the oldest URLs in cache to accommodate the newest URLs and their contents; and (d) upon receipt of the HTTP response to the HTTP request, the cache provides the response to the browser and stores the contents of the HTTP response in browser cache under the URL specified in the matching HTTP request. The IP packet of an HTTP request includes, among other things, a GET (or other permitted) command, the requested URL, the IP address and TCP port number (collectively called the "source socket") of the end-user's computer, and the destination IP address and TCP port number (collectively called the "destination socket") of the Webserver on which the URL resides.

The source socket of the HTTP request becomes the destination socket of the HTTP response, and the destination socket of the HTTP request becomes the source socket of the HTTP response.

An HTTP request causes the Webserver identified in the request message (i.e., the host computer at the domain name specified in the URL in the HTTP request) to send a Webpage (or Webpage component) identified by the file component of the URL to the requesting browser. The Webpage (or Webpage component) sent by the identified Webserver is the payload of the HTTP response message. "Browser" in this document means any end-user application software that uses HTTP. "HTTP" includes any equivalent or successor protocol to HTTP that performs "applications and services" functions at OSI layers 5 to 7 substantially similar to those of HTTP. "TCP/IP" includes any equivalent or successor protocol to TCP/IP that uses methods of message addressing, routing, transport, and headers that are substantially similar to those of TCP/IP.

In normal operation, a Web browser (or other software application that uses HTTP) only receives the contents of TCP/IP packets that contain (a) the destination IP address of the electronic device on which the browser is running, and (b) a TCP port number assigned to the Web browser application for the duration of the TCP connection by the operating system of the electronic device (or by other software controlling the allocation of TCP ports on the device). "Electronic device" or "end-user node" means herein any type of computational device (a) under the control of an operating system, such as a personal computer ("PC"), set-top box, or Internet appliance, and (b) that is interfaced with a dual channel or single channel network. An end-user node can be used by a human operator or be an unmanned processing node in a computer network. "End-user" means herein the client application (typically a browser) that originates HTTP requests and is the final recipient of HTTP responses. The most common network interfaces are network interface cards and dial-up networking adapters. In normal operation, a Web browser opens a TCP port, sends an HTTP request on that TCP port, and waits for an HTTP response on that TCP port. More accurately, the browser (or other client application software) requests that the operating system assign to the browser (or other client application software) an unused, unreserved TCP port number. The browser then uses that TCP port for a communications session. When the browser no longer needs the TCP port, the browser relinquishes the TCP port number (and therefore the TCP port the TCP port number identifies) back to the operating system, and the TCP port number can be reassigned to another application. Approximately sixty-four thousand TCP ports are available, which enables a browser to obtain multiple, concurrent TCP ports and to conduct multiple, concurrent HTTP sessions. Typically, the operating system assigns TCP port numbers in an ascending, sequential manner. When a TCP session is closed and the TCP port is released back to the operating

system, the same port number will normally not be used again until the operating system has cycled through the remaining (approximately 64,000) TCP ports. If the IP destination address of a packet available to a network interface of an electronic device does not match the (or an) IP address of the network interface, the network interface normally ignores or discards the packet. If the IP destination address does match the (or an) IP address of such network interface, the TCP/IP stack running in the electronic device forwards the contents of the packet to the application (e.g., browser) that opened the TCP port identified in the TCP packet header. If the specified TCP port is not open, the packet is ignored or discarded.

A traditional standalone cache (not part of a Web browser) matches an HTTP request with the HTTP response generated as a reply to the HTTP request, and stores the contents of the HTTP response under the URL specified in the HTTP request. The traditional standalone cache design requires collocation of the cache at a network access point where the cache can monitor and buffer traffic as required to provide the caching function. Being located at such a network access point permits a cache to monitor, copy and respond to HTTP requests and to store or cache HTTP responses that transit the access point. The caching device causes each HTTP request and each HTTP response to be stored in buffer memory and/or on a disk drive or other storage. When the URL is stored in the cache, the caching device accesses the buffered or cached information and responds to an HTTP request with the contents of the requested URL. If a URL is not stored in the cache, or when it is necessary to verify that the requested URL has not changed (which depends on specific cache configuration settings), the cache opens a TCP/IP connection to the designated Webserver (on a separate TCP port managed by the TCP/IP stack serving the cache) and returns an appropriate HTTP response to the originator of the HTTP request after receiving the HTTP response from the Webserver. With traditional caching devices, the Webserver receives HTTP requests with IP headers containing the IP address and TCP port numbers assigned by the caching device rather than those of the originating device, such as an end-user node.

The term "star topology, dual channel network" as used herein means satellite and terrestrial network architectures in which a plurality of transmit/receive ("TR") remote terminals share a return channel (called the outbound channel) transmitted from a central switching, or "hub", node to the remote terminals, and in which each TR remote terminal uses a uniquely assigned forward communications channel (called the inbound channel) that is transmitted from a TR remote terminal to the hub node and is not shared with (and is not received by) other remote terminals. Very small aperture terminal ("VSAT") satellite networks with transmit/receive ("two-way") remote terminals are an example of a star topology, dual channel network. In star topology, dual channel networks other than

two-way VSAT, the inbound channel from a TR remote terminal to a hub node serving the TR remote terminal can be through a terrestrial network, such as by use of terrestrial microwave, copper pair, or fiberoptic media; the outbound channel can also be through a terrestrial network, such as in digital cablecast networks. All TR remote terminals can communicate interactively with a hub node.

The term "star topology, single channel network" as used herein means satellite and terrestrial network architectures in which a plurality of receive-only ("RO") remote terminals share a return channel (called the "broadcast" or outbound channel) from a hub node, and in which each RO remote terminal lacks any inbound channel, direct or indirect, from an RO remote terminal to a hub node. VSAT satellite networks with receive-only ("one-way") remote terminals are a common example of a star topology, single channel network. A "star topology" network, without qualification as to channels, means star topology network with a population of TR remote terminals, RO remote terminals, or a mixed population of TR and RO remote terminals. A "remote terminal" means a device that uses radio frequency, electrical, photonic, or other communications technologies for one-way (RO) or two-way (TR) communications between that device and a hub node. A remote terminal can be interfaced with one or more local area networks, or equivalent connectivity, shared by end-user nodes and other types of nodes.

A traditional, standalone cache interfaced with a TR remote terminal in a star topology, dual channel network can monitor (a) only the locally originated, remote terminal-to-hub node, inbound channel, and (b) a hub node-to-remote terminals, outbound channel. The locally originated inbound channel contains HTTP requests from end-users interfaced with the remote terminal. The outbound channel contains HTTP responses to a plurality of remote terminals and the end-users interfaced with such remote terminals. A TR remote terminal in a star topology, dual channel VSAT network cannot directly monitor the inbound channels from other TR remote terminals to the hub node. An RO remote terminal in a star topology, single channel (broadcast) network does not have an inbound channel; there are no transmissions from the RO remote terminal to a hub node and therefore no inbound HTTP requests to monitor. An RO remote terminal also cannot directly monitor the inbound channels from other TR remote terminals to the hub node. An RO remote terminal, however, may be interfaced with a local area network on which local HTTP requests are carried; a cache associated with an RO remote terminal can provide HTTP responses to local HTTP requests. Assuming a TR or RO remote terminal could receive HTTP responses addressed to other remote terminals, except in extremely rare instances described below, a traditional, standalone cache associated with such remote terminal can not determine the URLs of such HTTP responses.

2. Description of Related Art

There are a great variety of caching technologies taught by prior art. Some caching methods are optimized for use in computer systems (U.S. Patent No. 5,930,515, issued to Ducharme, et al.), in Internet service provider operations (U.S. Patent No. 5,991,306, issued to Burns, et al.), or in satellite broadcasting (U.S. Patent No. 5,987,233, issued to Humphrey). The related art generally uses a "store and forward" approach in which files or data from a source device (e.g., a Webserver) are replicated on one or more other storage devices, or caches. The refresh frequency and selection of files and data is operator-defined or based on the actual or predicted frequency with which the files or data have been, or will be, requested by end users. Cache refresh can be either by the source initiating a refresh and "pushing" files and data out to cache storage, or by a destination initiating a refresh and "pulling" files and data from the source. Existing caching methods can be configured using various rules for refreshing cache contents, for using drive space, for blocking specified URLs, etc. Such caches can also be configured for use as proxy servers, in which case an HTTP request generated by an end-user's browser uses a destination TCP port defined for the proxy server rather than using a well-known TCP port defined for public Webservers. TCP port 80 is a well-known port used by public Webservers for receiving HTTP requests. Caches can be cascaded, in which case each cache in series either contains the requested URL and returns the appropriate HTTP response to the requesting browser, or opens a TCP/IP connection to the next cache in series and sends an HTTP request to the next cache for the URL. In a cascaded cache architecture, when a cache does not contain a requested URL, the cache generates an HTTP request to the next device in series, and upon receipt of the HTTP response (which may come from the next or subsequent cascaded caching device, or from the Webserver at the end of the cascade), forwards the HTTP response to the end-user, or to a previous cache in a cascaded series of caches.

HTTP versions prior to HTTP 1.1 did not provide a field for the URL in the header of an HTTP response. The implication of this is that a device that monitors HTTP responses will not have access to the URL of the payload of an HTTP response when earlier versions of HTTP are used. This being the case, the URL and its contents cannot be cached when the HTTP request that generated the HTTP response is not available. When HTTP version 1.1 or later is used, the Content Location header (a field reserved in an HTTP response for the URL of the payload of the response) is an optional field, and Webserver programmers are not required to include the field in the HTTP response. The use of the Content Location header is extremely rare; even when the Content Location header is included, the URL field may be missing or incorrect. When programmers do not include this field or the stated URL is missing or incorrect, an HTTP response cannot be

matched to the corresponding URL unless the HTTP request is available.

None of the caching technologies described in the related art addresses the problem of a remote terminal in a star topology network not having direct access to the inbound channels of other remote terminals, or of the omission of URLs in HTTP responses in outbound channels. Traditional, standalone caching technologies for star topology, dual channel networks, such as that described in U.S. Patent No. 5,987,233, issued to Humphrey, require that a "global" standalone cache be maintained at one or more hub nodes (e.g., the hub earth station of a VSAT network) and that a remote terminal send HTTP requests through an inbound channel for all URLs that are not in the built-in browser cache, or other local cache, on a node interfaced with a remote terminal. The purpose of traditional caches, especially those associated with a hub node, is to reduce the volume of HTTP requests and HTTP responses transiting the span between the cache and the Internet, and therefore reduce the required datarate of (or increase the number of users supported over) such span. FIG.1 depicts the current system of caching in a star topology, dual channel network. FIG 1A illustrates message exchange when a URL is in a local cache. FIG 1B illustrates message exchange when a URL is not in local cache or in a traditional hub cache. In addition, the use of a traditional cache at a hub node of a satellite network provides a proxy function that "intercepts" end-users' HTTP requests for URLs, responds with those URLs cached at the hub node (which avoids Internet propagation delays), but does nothing to avoid the deterioration of response time due to satellite propagation delays between the end-user's node and the hub node. The contents of a "global" cache at a hub node reflect the activities of all end-users served by the hub node(s). In a multi-hub architecture, a "central" or "master" hub periodically collects the cache content from all other hub caches by pulling the contents from the hubs and creating an aggregate cache. The aggregate cache is then periodically pushed to the other hubs to replicate the aggregate cache at all hubs. In this architecture, the hub locations are generally terrestrial hubs, e.g. an internet service provider point of presence, with end-users supported via terrestrial (usually dial-up) access to the hub node. Locating a global cache at a hub node has been necessary because only a hub node provides a network access point that permits the monitoring of all HTTP requests from, and corresponding HTTP responses to, all remote terminals. Such caches generate, or obtain and relay, HTTP responses on behalf of client applications running on end-user nodes. Providing global caches at intermediate nodes or at end-user nodes in a star topology network has heretofore required that the cache contents be actively pushed or pulled from the "master caching center" or global cache at the hub node(s). An "intermediate node" is a node between the hub node and an end-user node in a star topology network. Accordingly, there is a need for a more efficient and cost effective method and system of building a global cache at intermediate

nodes and at end-user nodes in star topology networks.

The Passive Internet Cache system overcomes the limitation of having to collocate a global cache with the hub node(s) of a star topology network, and the limitation imposed by the omission of the URL in HTTP responses. The Passive Internet Cache system uses an innovative, "passive listening" method to construct a global cache at any intermediate node, at a series of cascaded intermediate nodes, or at end-user node in star topology networks. The Passive Internet Cache system's provision of an HTTP response from a global cache at or very near an end-user node means higher cache "hit" rates for the end-user, faster response times, and a corresponding reduction in the use of inbound and outbound communication channels.

SUMMARY OF THE INVENTION

The Passive Internet Cache ("PIC") system meets the need for a more efficient and cost effective global cache in intermediate nodes or end-user nodes in star topology networks. The PIC system builds a global cache at or near end-user nodes, and substantially avoids response delays at end-user nodes arising from the exchange of request and response messages between a remote terminal and a hub node. Specifically, the PIC system operates in the background and enables end-user nodes interfaced with remote terminals to use the contents of all HTTP responses in an outbound channel, reduces the repetitive transmission of the same HTTP response payloads over an outbound channel, and in dual channel networks also reduces the need to send HTTP requests over an inbound channel. The PIC system can be used and is effective in any type of star topology network. Suitable dual channel networks include VSAT systems with inbound and outbound channels by satellite, or with outbound channel by satellite and inbound channel by terrestrial network; Direct-to-Home ("DTH") satellite systems that have a broadcast outbound satellite channel and a terrestrial inbound channel (dial-up, leased line, cable modem, etc.); and terrestrial systems such as multichannel multipoint distribution service ("MMDS"), local multipoint distribution service ("LMDS"), instructional television fixed service ("ITFS"), digital television broadcasting, and digital cablecasting systems that have a terrestrial "broadcast" or "multicast" outbound channel and various types of inbound channels. "Broadcast" means a single message in the outbound channel can be addressed to and received by all remote terminals. "Multicast" means a single message in the outbound channel can be addressed to and received by a portion of, but not all, remote terminals. In addition to star topology, dual channel networks, the PIC system is effective in networks with single channel, receive only terminals, such as receive-only VSAT and DTH satellite terminals. The PIC system is also effective with MMDS, LMDS, ITFS, digital television broadcasting, and digital cablecasting terminals that may have two-way (transmit/receive) or one-way (receive only) capabilities, whether wireline and/or wireless.

A PIC system can be configured to serve a mixed population of TR remote terminals and RO remote terminals, and can be configured to use concurrently wireline outbound channels, wireless outbound channels, wireline inbound channels, and/or wireless inbound channels. This document assumes that end-users are associated with remote terminals and originate HTTP requests, and that Webservers (which may be on a local area network at the hub node, or may be accessed through other networks serving the hub node) provide HTTP responses to the hub node, and the hub node transmits such HTTP responses in the outbound channel to the remote terminals.

The Passive Internet Cache system has five primary components: a monitoring component, an encapsulating component, a buffering component, an unencapsulating ("decapsulating") component, and a global cache (called a "PIC Webcache"). A primary monitoring component and an encapsulation component are implemented at a hub node. A buffering component can be implemented at a hub node, intermediate node, and/or end-user node. The decapsulating component and the PIC Webcache are implemented at an intermediate node or at an end-user node of a star topology network. The decapsulating component is interfaced or integral with a buffering component at an intermediate node or at an end-user node. An example of an intermediate node is a computer or set-top box interfaced with, or integral with, a VSAT, DTH, MMDS, LMDS, ITFS, digital television broadcasting, or digital cablecasting remote terminal and also interfaced with one or more end-user nodes; the interfaces can be a local area network, a system bus, universal serial bus ("USB"), or similar connectivity. The distribution of the components of a PIC system defines the various embodiments of the invention. All embodiments of the PIC system require that a remote terminal output all HTTP responses received in the outbound channel, not just those HTTP responses with IP addresses assigned to intermediate nodes and end-user nodes interfaced with such remote terminal, to each instance of a secondary monitoring component associated with or interfaced with such remote terminal. Such output can be obtained by setting configuration options with some commonly available remote terminal or hub node systems. If a remote terminal and hub node system does not support reception of HTTP responses for nodes other than those attached to a given remote terminal, additional hardware and/or software components, such as a signal splitter and separate receiver/decoder, must be added to provide the ability to receive all HTTP responses in an outbound channel at a remote terminal.

One embodiment of the PIC system, in the context of a star topology, dual channel VSAT network, implements a primary monitoring component and an encapsulating component at a hub earth station, and implements a secondary monitoring component, a decapsulating component, a buffering component, and a PIC Webcache on an end-user node interfaced with a VSAT remote terminal. On the end-user node, the PIC Webcache

is configured as a proxy server cache and the Web browser is configured to use such proxy server cache. The primary monitoring component of the PIC system at the hub earth station monitors all inbound channels from all the VSAT remote terminals to the hub earth station. The primary monitoring component copies all HTTP requests from the monitored inbound channels, forwards the HTTP requests to the encapsulating component at the hub earth station where the HTTP requests are immediately encapsulated in a User Datagram Protocol ("UDP") message. Such UDP messages are then immediately transmitted in the outbound channel using a broadcast IP address and UDP port monitored by all remote terminals that receive such outbound channel and are interfaced with an end-user node that has a PIC Webcache. The secondary monitoring component at the end-user node communicates through a network interface on the end-user node with the VSAT remote terminal. The secondary monitoring component at the end-user node also communicates with the decapsulation component and with the buffering component on the end-user node. The secondary monitoring component copies from the VSAT remote terminal to the decapsulation component all UDP messages, and to the buffering component all HTTP messages, received by the remote terminal, not just those packets with an IP destination address of the end-user node. When the secondary monitoring component on the end-user node receives a UDP packet containing an encapsulated HTTP request, it provides such packet to the decapsulation component. The decapsulation component removes the UDP encapsulation ("decapsulation"), and forwards the decapsulated HTTP request to the buffering component on the end-user node. The buffering component on the end-user node communicates with disk drive(s) or other read/write storage device(s) associated with the end user node that are used by the PIC Webcache for storage. The buffering component at the end-user node stores the decapsulated HTTP request in buffer memory, and opens a file (or database record) tagged with the source socket (source IP address and TCP port number) and the destination socket (destination IP address and TCP port number) specified in the decapsulated HTTP request. The tagged file (or database record) awaits the arrival of the relevant HTTP response to be delivered later by the buffering component.

Upon receipt at the hub earth station of an HTTP response from a Webserver, the hub earth station transmits the HTTP response in the outbound channel. If a browser running on a end-user node interfaced with a VSAT remote terminal originated the HTTP request that generated the HTTP response, the HTTP response is received in the outbound channel by that VSAT remote terminal, forwarded to the PIC Webcache in the normal manner, and delivered by the PIC Webcache, operating as a proxy server cache, to the requesting browser. The secondary monitoring component at the end-user node also forwards every HTTP response in the outbound channel to the buffering component running on the end-user node. If a browser running on an end-user node interfaced with

a VSAT remote terminal did not originate the HTTP request that generated the HTTP response, the contents of the HTTP response are not passed to the browser upon receipt. Rather, the buffering component examines the HTTP response to determine the source and destination sockets and stores the contents of the HTTP response in the file (or database record) previously tagged with those sockets (but with source and destination sockets reversed in the HTTP request as compared with the HTTP response). After receiving both the HTTP request and the HTTP response, the buffering component at the end-user node can match the HTTP request with the correct HTTP response, extract the URL identity from the HTTP request, and create a file (or database record) for the URL on the storage device(s) used by the PIC Webcache as if the browser itself had caused the PIC Webcache to cache the URL and its contents, but without displaying the URL contents and without issuing an HTTP request. To receive and store the HTTP response, the PIC system does not obtain or use a TCP port number allocated by the end-user node or interfere with normal browser operation. If the browser requests a PIC-cached URL in the future, the contents of the URL will already be in the PIC Webcache, and will be delivered instantly from the PIC Webcache to the browser.

In summary, the Passive Internet Cache system without buffering at a hub node comprises a hub node of a star topology network that receives HTTP requests from one or more remote terminals, which hub node is interfaced with at least one external server computer and transmits UDP datagrams, HTTP requests, and HTTP responses in an outbound channel from said hub node of the star topology network, and one or more intermediate or end-user nodes of said network that receive said UDP datagrams, HTTP requests, and HTTP responses and which one or more intermediate or end-user nodes are associated with or interfaced with a means for socket-matching HTTP requests and HTTP responses and for storing in an end-user accessible cache the contents of a given HTTP response under the URL extracted from a correctly matched HTTP request.

The Passive Internet Cache system with buffering at a hub node comprises a hub node of a star topology network that receives HTTP requests from one or more remote terminals, which hub node is interfaced with at least one external server computer, has a first means for socket-matching, and transmits UDP datagrams, HTTP requests, and HTTP responses in an outbound channel from said hub node of the star topology network, and one or more intermediate or end-user nodes of said network that receive said UDP datagrams, HTTP requests, and HTTP responses and which one or more intermediate or end-user nodes are associated with or interfaced with a second means for socket-matching HTTP requests and HTTP responses and for storing in an end-user accessible cache the contents of a given HTTP response under the URL extracted from a correctly matched HTTP request.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other aspects and features of the Passive Internet Cache system will be better understood from the following more detailed description and appended claims in conjunction with the drawings, of which:

FIG. 1 illustrates the current system of caching in a star topology, dual channel network.

FIG 1A illustrates message exchange when a URL is in a local cache.

FIG 1B illustrates message exchange when a URL is not in local or hub cache.

FIG. 2 illustrates the Passive Internet Cache embodiment with socket matching and PIC Webcache at an end-user node in a star topology, dual channel network (first embodiment).

FIG. 2A illustrates the distribution of PIC components in the first embodiment.

FIG. 2B illustrates passive listening and encapsulation at a hub node.

FIG. 2C illustrates socket matching.

FIG. 2D illustrates message exchange using socket matching to store URLs and URL contents in a PIC Webcache in the first embodiment.

FIG. 3 illustrates the Passive Internet Cache embodiment with socket matching at an intermediate node and with PIC Webcache at an end-user node in a star topology, dual channel network (second embodiment).

FIG. 3A illustrates the distribution of PIC components in the second embodiment.

FIG. 4 illustrates the Passive Internet Cache embodiment with socket matching and PIC Webcache at an intermediate node in a star topology, dual channel network (third embodiment).

FIG. 4A illustrates the distribution of PIC components in the third embodiment.

FIG. 5 illustrates the Passive Internet Cache embodiment with socket matching at the hub node and with secondary buffering and PIC Webcache at an end-user node in a star topology, dual channel network (fourth embodiment).

FIG. 5A illustrates the distribution of PIC components in the fourth embodiment.

FIG. 6 illustrates the Passive Internet Cache embodiment with socket matching at the hub node, with secondary buffering at an intermediate node, and with PIC Webcache at an end-user node in a star topology, dual channel network (fifth embodiment).

FIG. 6A illustrates the distribution of PIC components in the fifth embodiment.

FIG. 7 illustrates the Passive Internet Cache embodiment with socket matching at the hub node and with PIC Webcache at an intermediate node in a star topology, dual channel network (sixth embodiment).

FIG. 7A illustrates the distribution of PIC components in the sixth embodiment.

FIG. 8 illustrates the Passive Internet Cache embodiment with socket matching and

PIC Webcache at an end-user node in a star topology, single channel network (seventh embodiment).

FIG. 8A illustrates the distribution of PIC components in the seventh embodiment.

FIG. 9 illustrates the Passive Internet Cache embodiment with socket matching at an intermediate node and with PIC Webcache at an end-user node in a star topology, single channel network (eighth embodiment).

FIG. 9A illustrates the distribution of PIC components in the eighth embodiment.

FIG. 10 illustrates the Passive Internet Cache embodiment with socket matching and PIC Webcache at an intermediate node in a star topology, single channel network (ninth embodiment).

FIG. 10A illustrates the distribution of PIC components in the ninth embodiment.

FIG. 11 illustrates the Passive Internet Cache embodiment with socket matching at the hub node and with secondary buffering and PIC Webcache at an end-user node in a star topology, single channel network (tenth embodiment).

FIG. 11A illustrates the distribution of PIC components in the tenth embodiment.

FIG. 12 illustrates the Passive Internet Cache embodiment with socket matching at the hub node, with secondary buffering at an intermediate node, and with PIC Webcache at an end-user node in a star topology, single channel network (eleventh embodiment).

FIG. 12A illustrates the distribution of PIC components in the eleventh embodiment.

FIG. 13 illustrates the Passive Internet Cache embodiment with socket matching at the hub node and with secondary buffering and PIC Webcache and at an intermediate node in a star topology, single channel network (twelfth embodiment).

FIG. 13A illustrates the distribution of PIC components in the twelfth embodiment.

FIG. 14 is a block diagram of intermediate node and end-user node functional modules in a star topology, dual channel network.

FIG. 14A is a block diagram of hub node functional modules node in a star topology, dual channel network.

FIG. 15 is a block diagram of intermediate node and end-user node functional modules in a star topology, single channel network.

FIG. 15A is a block diagram of hub node functional modules node in a star topology, single channel network.

FIG. 16 is a block diagram of a PIC system with content authorization.

FIG. 17 is a block diagram of error correction by a buffering component based on continuity of TCP sequence numbers.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Preferred embodiments of the invention will now be described with reference to FIGS. 1 to 17. It will be appreciated by those of ordinary skill in the art that the description given herein with respect to those figures is for exemplary purposes only and is not intended in any way to limit the scope of the invention.

The Passive Internet Cache has five components, some of which may be distributed among hub, intermediate, and/or end-user nodes, as described below. The five component are: a monitoring component, an encapsulating component, a buffering component, an unencapsulating ("decapsulating") component, and a global cache (called a "PIC Webcache"). A primary monitoring component and an encapsulation component are implemented at hub node(s) in all embodiments. A buffering component can be implemented at a hub node, intermediate node, and/or end-user node. The primary monitoring component at a hub node monitors all IP addresses and TCP ports involved in HTTP sessions in use by all nodes using inbound and outbound channels handled by such hub node, and provides those addresses and ports to the various implemented buffering component(s) to enable tracking of all IP addresses and TCP ports involved in HTTP sessions; such buffering and tracking enables the matching of source sockets and destination sockets used in each of the embodiments. The encapsulating component encapsulates HTTP requests in broadcast UDP (or other connectionless protocol suitable for broadcast or multicast transmission) messages that are transmitted from a hub node through an outbound channel and are addressed to all remote terminals that can receive that outbound channel. A secondary monitoring component, a decapsulating component, a buffering component, and a PIC Webcache can be implemented at an intermediate node or at an end-user node. The decapsulating component is interfaced or integral with a buffering component at an intermediate node or at an end-user node. An example of an intermediate node is a computer or set-top box interfaced with, or integral with, a remote terminal and also interfaced with one or more end-user nodes; the interfaces can be a local area network, a system bus, USB, or similar connectivity. "Associated with" means two hardware or software devices are either integral or very closely interfaced, as through a system bus, proximate area network, USB, high speed local area network, or other close range connectivity, for message exchange. "Interfaced with" means two hardware or software devices are closer in distance to each other than to the hub node, normally have a higher datarate connection between such devices than to the hub node, and are interconnected through a local area network, or other medium range connectivity, for message exchange. "Buffer" or "buffer memory" means random access memory, disk storage, or equivalent read/write technology suitable for the short term storage and retrieval of digital

information.

With reference to FIG. 2B, passive listening and encapsulation is implemented at a hub node of a dual channel, star topology network by providing a primary monitoring component (201) with unlimited, or "promiscuous," access to all HTTP messages inbound from remote terminals. A hub node in such a star topology network typically has a plurality of remote terminals whose inbound channels (each denoted "Rcv" in FIG. 2B) are received at the hub node, and whose TCP/IP messages are routed to the Internet, a private network, and/or a traditional cache. The primary monitoring component (201) listens to all inbound HTTP messages in such TCP/IP messages, and provides HTTP requests to the encapsulation component (202). The encapsulation component (202) encapsulates each HTTP request as a broadcast UDP (or other connectionless protocol suitable for broadcast or multicast transmission) message and provides each such message to the outbound channel of the hub node (denoted as "Xmit" in FIG. 2B).

The distribution of the components of a PIC system within a star topology network defines the various embodiments of the invention. All embodiments of the PIC system require that a remote terminal output all HTTP responses received through the outbound channel, not just those HTTP responses with IP addresses assigned to intermediate nodes and end-user nodes interfaced with such remote terminal, to each instance of a secondary monitoring component associated with or interfaced with such remote terminal. Such output can be obtained by setting configuration options with some commonly available remote terminal or hub node systems. In other instances, existing remote terminal and hub node systems may not support reception of HTTP responses by a remote terminal other than HTTP responses addressed to a specific remote terminal; in these cases, additional hardware and/or software components must be added to a remote terminal to provide the ability to receive all HTTP responses in an outbound channel. Where such additional hardware and software components are required to receive all HTTP responses in an outbound channel, a signal splitter (e.g., a RF- or IF-band splitter for wireless outbound channels) and separate receiver/decoder installed at a remote terminal is usually sufficient.

The HTTP requests and HTTP responses transmitted through the communications paths in a PIC system are not handled through a traditional TCP/IP stack and do not involve the use of TCP acknowledgements. The buffering component, whether associated with or interfaced with a hub node, an intermediate node, or an end-user node, uses socket-matching to pair an HTTP request with the correct HTTP response. With reference to FIGS. 2 and 2C, in the socket-matching process in a PIC embodiment with buffering and PIC Webcache at an end-user node, the buffering component (206) examines the packets forwarded by the secondary monitoring component (204). The field sequence observed

in HTTP packets is: source IP address (abbreviated "IPs-" in FIG. 2C), destination IP address (abbreviated "IPd-" in FIG. 2C), source TCP port number (abbreviated "TCPs-" in FIG. 2C), destination TCP port number (abbreviated "TCPd-" in FIG. 2C), and the payload (either an HTTP request or an HTTP response). The secondary monitoring component (204) forwards each HTTP response (the first packet in packet stream 220, in FIG 2C) directly to the buffering component (206) and forwards each UDP packet (the second packet in packet stream 220, in FIG 2C) to the decapsulation component (205, in FIG. 2), which decapsulates any HTTP request contained in such packet and forwards the HTTP request so decapsulated to the buffering component (206). When the buffering component (206) receives an HTTP response, it examines the source and destination sockets, and checks to see if a file (or database record) in buffer memory containing an HTTP request has been tagged with those source and destination sockets. The source and destination sockets are reversed in an HTTP request as compared with the corresponding HTTP response. The buffering component compensates for the reversal of source and destination sockets when it compares HTTP requests and HTTP responses. If the buffering component can match a received HTTP response with a buffered HTTP request, it extracts the URL identity from the HTTP request, and creates a file (or database record) for the URL on the storage device(s) (212, in FIG. 2) used by the PIC Webcache (207, in FIG. 2) as if the PIC Webcache itself had cached the URL. If the buffering component cannot match the a received HTTP response with a buffered HTTP request, the buffering component creates a file (or database record) tagged with the sockets of the HTTP response, and stores such file (or database record) in buffer memory. In the case of packet stream (220), the HTTP response with addressing of "IPsb, IPda, TCPsb, TCPda," in the first portion of packet stream (220) would be buffered and subsequently matched with the decapsulated HTTP request with addressing of "IPsa, IPdb, TCPsa, TCPdb," in the second portion of packet stream (220). To introduce a second example of socket matching, if the buffering component (206) receives the HTTP request in the first packet in packet stream (221) that contains source and destination sockets that the buffering component cannot match with a previously buffered HTTP response, the buffering component creates a file (or database record) tagged with the sockets of that HTTP request, and buffers such file (or database record). When the buffering component subsequently receives the HTTP response in the second portion of packet stream (221), it examines the source and destination sockets of that HTTP response, matches those sockets with the previously buffered file (or database record) tagged with those source and destination sockets (compensating for the reversal of source and destination sockets), extracts the URL identity from the HTTP request, and creates a file (or database record) for the URL on the storage device(s) (212) used by the PIC Webcache (207) as if the PIC Webcache itself had cached

the URL. Regardless of whether the HTTP request or the HTTP response is received first, to receive and store the URL and the contents of the URL, the PIC system does not obtain or use a TCP port number allocated by the end-user node. Thus, even though the browser at the end-user node had not previously requested a URL in the PIC Webcache, if the browser does request that URL in the future, the contents of the URL will already be in the PIC Webcache serving the browser, and will be delivered instantly from the PIC Webcache to the browser.

FIG. 2D illustrates the messages exchanged in the operation of a PIC Webcache in a PIC embodiment with buffering and PIC Webcache at an end-user node. In FIG. 2D, an unseen remote terminal transmits an HTTP request that is monitored by the primary monitoring component at a hub node and handled as described above and as depicted in the upper, leftward message flows in FIG. 2D. That HTTP request is also sent in the customary manner from the hub node to a traditional proxy or cache as depicted in the upper, rightward message flows in FIG. 2D; if the URL is not found in the traditional proxy or cache, the HTTP request is forwarded to the Webserver identified in the URL. The traditional proxy or cache, or the Webserver, as the case may be, replies to the HTTP request with an HTTP response, which the hub node transmits in the outbound channel as described above and as depicted in the lower, leftward message flows in FIG. 2D. Socket-matching and storage of the URL and its contents is performed at the end-user node, as described above.

The buffering component in the first embodiment only proceeds with writing the contents of the HTTP response in the PIC Webcache under the URL contained in the HTTP request if no errors are detected in the HTTP response and HTTP request. The buffering component uses the error detection techniques of TCP, such as confirming a continuous series of TCP header sequence numbers, to detect errors. If no errors are detected, the buffering component writes the contents of the matched HTTP response in the PIC Webcache under the URL contained in the matching HTTP request. If errors are detected in an HTTP request, the request is discarded, and any HTTP responses that have not been matched after the expiration of an operator-controlled time period are also discarded. If errors are detected in an HTTP response, and the intermediate node or end-user node has access to an inbound channel, the buffering component can issue an HTTP request for a replacement HTTP response. If errors are detected in the HTTP response, and the intermediate node or end-user node does not have access to an inbound channel by which to issue an HTTP request, the matched but defective HTTP response and matching HTTP request are discarded. Normally, any single end-user node would only request a small portion of the contents of a PIC Webcache within a given time period, so such discarding of corrupted or missing HTTP responses and HTTP requests should have an

insignificant impact on the overall performance improvements afforded by use of a PIC Webcache.

The embodiments of the PIC system fall into two categories: embodiments for dual channel, star topology networks, and embodiments for single channel, star topology networks. With each category, there are two sub-categories of embodiments: embodiments that perform initial socket-matching at a hub node, and embodiments that that do not perform any socket-matching at a hub node. FIGS. 14 and 14A are block diagrams of intermediate node, end-user node, and hub node functional modules in a star topology, dual channel network. FIGS. 15 and 15A are block diagrams of intermediate node, end-user node, and hub node functional modules in a star topology, single channel network. Within each sub-category, there are three basic embodiments that are distinguished by whether a buffering component is implemented at an intermediate node or at an end-user node, and by whether a PIC Webcache is implemented at an intermediate node or at an end-user node, as illustrated in FIGS. 14 and 15 for dual channel and single channel networks, respectively.

First Embodiment: Buffering and PIC Webcache at end-user node (TR: Socket Matching at End-User Node ("EUN"))

FIG. 2 illustrates the Passive Internet Cache embodiment with socket matching and PIC Webcache at an end-user node in a star topology, dual channel network. FIG. 2A illustrates the distribution of PIC components in the first embodiment. FIG. 2B illustrates passive listening and encapsulation. FIG. 2C illustrates socket matching. FIG. 2D illustrates message exchange using socket matching to store URLs and URL contents in a PIC Webcache in the first embodiment. Reference numerals for the first embodiment are to Fig. 2 unless otherwise noted.

A first embodiment of the Passive Internet Cache system in a star topology, dual channel network implements a primary monitoring component (201) and an encapsulation component (202) at a hub node (203 in Fig. 2A), and a secondary monitoring component (204), a decapsulation component (205), a buffering component (206), and a PIC Webcache (207) at an end-user node (208 in Fig. 2A) interfaced with a remote terminal (209). In the context of a star topology, dual channel network, this embodiment of the Passive Internet Cache system provides the lowest cost end-user node for a PIC system. Assuming a remote terminal can output all HTTP responses by the use of configuration settings, the typical PIC components at the end-user node are software modules installed on an end-user PC and, as such, require no hardware investment to implement. The PIC Webcache is configured as a proxy server cache and the Web browser on the end-user node is configured to use such proxy server cache. The primary monitoring component (201) at the hub node (203) monitors all inbound channels transmitted from all the remote

terminals (209, 213) to the hub node (203). The primary monitoring component (201) copies all HTTP requests from the monitored inbound channels (214, 215), provides each HTTP request to the encapsulating component (202), and the encapsulating component (202) immediately encapsulates each copied HTTP request in a UDP (or other connectionless protocol suitable for broadcast or multicast transmission) message. The encapsulating component provides each UDP message to the outbound channel (216) for transmission in the outbound channel using a broadcast IP address and UDP port (or other suitable protocol, address, and port) monitored by all remote terminals that receive such outbound channel and are interfaced with PIC components on end-user nodes. The secondary monitoring component (204) on the end-user node (208) communicates through a network interface on the end-user node with the remote terminal (209). Using a first communications path (210) on the end-user node (208), the secondary monitoring component (204) also communicates with the decapsulation component (205), and the decapsulation component (205) communicates with the buffering component (206). The secondary monitoring component (204) copies all of the encapsulated UDP packets from the remote terminal (209) to the decapsulation component (205) through a first portion of the first communications path (210). When the secondary monitoring component (204) on the end-user node provides a UDP packet containing an encapsulated HTTP request to the decapsulation component (205), the decapsulation component (205) removes the UDP encapsulation ("decapsulation"), and forwards each decapsulated HTTP request to the buffering component (206) at the end-user node (208 in FIG. 2A) through a second portion of the first communications path (210). The buffering component (206) at the end-user node communicates with disk drive(s) (212) or other read/write storage device(s) associated with the end user node that are used by the PIC Webcache (207) for storage. The buffering component (206) stores the decapsulated HTTP request in buffer memory, and opens a file (or database record) tagged with the source socket (source IP address and TCP port number) and the destination socket (destination IP address and TCP port number) specified in the decapsulated HTTP request. The tagged file (or database record) would normally be on the storage device(s) used by the PIC Webcache. The tagged file (or database record) awaits the arrival of the contents of the relevant HTTP response to be delivered later from the buffering component.

Upon receipt at the hub node of an HTTP response, the hub node (203 in FIG. 2A) transmits the HTTP response in the outbound broadcast channel (216 in FIG. 2). If a browser running on a end-user node interfaced with a remote terminal originated the HTTP request that generated the HTTP response, the HTTP response is received in the outbound channel by that remote terminal, forwarded to the PIC Webcache in the normal manner, and delivered by the PIC Webcache, operating as a proxy server cache,

to the requesting browser. The secondary monitoring component (204) forwards through a second communications path (211) to the buffering component on the end-user node all HTTP packets received by the remote terminal (209), not just those packets with an IP destination address of the end-user node (208). If a browser running on an end-user node interfaced with a remote terminal did not originate the HTTP request that generated the HTTP response, the contents of the HTTP response are not passed to the browser upon receipt. Rather, the buffering component on the end-user node examines the HTTP response to determine the source and destination sockets and stores the contents of the HTTP response in the file (or database record) previously tagged with those sockets (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response). After receiving both the HTTP request and the HTTP response, the buffering component at the end-user node can match the HTTP request with the correct HTTP response, extract the URL identity from the HTTP request, and create a file (or database record) for the URL on the storage device(s) (212) used by the PIC Webcache as if the PIC Webcache itself had cached the URL. To receive and store the HTTP response, the PIC system does not obtain or use a TCP port number allocated by the end-user node. If the browser requests that URL in the future, the contents of the URL will already be in the PIC Webcache serving the browser, and will be delivered instantly from the PIC Webcache to the browser.

In the event the browser running on an end-user node interfaced with a remote terminal did not originate the HTTP request that generated an HTTP response provided by the monitoring component at the end-user node to the buffering component at the end-user node, and the buffering component at the end-user node has not yet received, via a UDP message from the hub node, a copy of the HTTP request that triggered such HTTP response, the buffering component at the end-user node stores the HTTP response in buffer memory, tagged by source socket and destination socket, until the associated HTTP request arrives via a UDP message from the hub node. Upon receipt and decapsulation of the HTTP request, the buffering component can match the HTTP request with the correct HTTP response, extract the URL identity from the HTTP request, and create a file (or database record) for the URL on the storage device(s) used by the PIC Webcache as if the PIC Webcache itself had cached the URL and its contents.

Second Embodiment: Buffering at intermediate node and PIC Webcache at end-user node (TR: Socket Matching at intermediate node ("IN"))

FIG. 3 illustrates the Passive Internet Cache embodiment with socket matching at an intermediate node and with PIC Webcache at an end-user node in a star topology, dual channel network. FIG. 3A illustrates the distribution of PIC components in the second embodiment. Reference numerals are to FIG. 3 unless otherwise noted.

A second embodiment of the Passive Internet Cache system in a star topology, dual channel network implements a buffering component (306) at an intermediate node (317 in FIGS. 3 and 3A) and a PIC Webcache (307) at an end-user node (308 in FIGS. 3 and 3A. The second embodiment has the advantage of requiring only one instance of a buffering component interfaced with a given remote terminal; the buffering component serves all end-user nodes on one or more local area networks (or other local connectivity) served by that remote terminal. This embodiment provides fast end-user response times through the use of PIC Webcaches at end-user nodes. Additionally, the use of a separate processor for the monitoring and buffering components at an intermediate node permits higher performance memory and a dedicated processor or processors to handle these functions, off-loading those processing requirements from the end-user nodes. TCP/IP is used in the local area network (or other connectivity) between the buffering component at the intermediate node and the PIC Webcache in end-user nodes.

The second embodiment of the Passive Internet Cache system implements a primary monitoring component (301) and an encapsulation component (302) at a hub node, a secondary monitoring component (304) , a decapsulation component (305), and a buffering component (306) at a computer or set-top box that is interfaced with, or is an integral part of, a remote terminal (309) and is also interfaced with a local area network (318) (or other connectivity) that serves one or more end-user nodes, and the PIC Webcache (307) component at one or more end-user nodes (308). The end-user nodes may also communicate directly with a remote terminal for communications unrelated to the PIC system. The PIC Webcache in this embodiment is configured as a proxy server cache and the Web browser on the end-user node is configured to use such proxy server cache. The primary monitoring component (301) and the encapsulation component (302) of the PIC system at the hub node (303) function as described above. The secondary monitoring component (304) at the intermediate node communicates through a network, internal bus or other interface with the remote terminal. The secondary monitoring component (304) also communicates with the decapsulation component (305), and with the buffering component (306), at the intermediate node. The secondary monitoring component (304) copies from the remote terminal (309) to the decapsulation component (305) on the intermediate node all encapsulated UDP packets, and to the buffering component (306) on the intermediate node, all HTTP packets, received by the remote terminal, not just those packets with an IP destination address of the intermediate node or of end-user nodes served by that remote terminal. When the secondary monitoring component (304) receives a UDP packet containing an encapsulated HTTP request, it provides such packet through a first portion of a first communications path (310) to the decapsulation component (305) at the intermediate node. The decapsulation component (305) decapsulates the HTTP

request, and forwards each decapsulated HTTP request through a second portion of the first communications path (310) to the buffering component (306) at the intermediate node. The buffering component (306) at the intermediate node communicates with disk drive(s) or other read/write storage device(s) (312) associated with each end user node that are used by one or more end-user nodes' PIC Webcaches (307) for storage. The buffering component at the intermediate node stores the decapsulated HTTP request in buffer memory, and opens a file (or database record) tagged with the source socket (source IP address and TCP port number) and the destination socket (destination IP address and TCP port number) specified in the decapsulated HTTP request. The tagged file (or database record) would normally be on the storage device(s) used by each PIC Webcache interfaced with the intermediate node. The tagged file awaits the arrival of the contents of the relevant HTTP response to be delivered later by the buffering component.

Upon receipt at the hub node of an HTTP response, the hub node (303 in FIG 3A) transmits the HTTP response in the outbound broadcast channel (316 in FIG. 3). If a browser running on a end-user node interfaced with a remote terminal originated the HTTP request that generated the HTTP response, the HTTP response is received in the outbound channel by that remote terminal, forwarded to the PIC Webcache in the normal manner, and delivered by the PIC Webcache, operating as a proxy server cache, to the requesting browser. The secondary monitoring component at the intermediate node forwards through a second communications path (311) every HTTP response in the outbound channel to the buffering component running on the intermediate node. If a browser running on an end-user node interfaced with a remote terminal did not originate the HTTP request that generated the HTTP response, the contents of the HTTP response are not passed to the browser upon receipt. Rather, the buffering component on the intermediate node examines the HTTP response to determine the source and destination sockets and stores the contents of the HTTP response in the file (or database record) previously tagged with those sockets (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response). After receiving both the HTTP request and the HTTP response, the buffering component at the intermediate node can match the HTTP request with the correct HTTP response, extract the URL identity from the HTTP request, and create a file (or database record) for the URL on the storage device(s) used by each PIC Webcache as if each PIC Webcache itself had cached the URL. To receive and store the HTTP response, the PIC system typically uses a local area network (318) and may use TCP port numbers allocated by end-user nodes interfaced with the intermediate node for the purpose of directly accessing and updating the PIC Webcache on the disk(s) of the end-user PCs. If the browser requests that URL in the future, the contents of the URL will already be in the PIC Webcache serving the browser, and will be delivered instantly from the PIC Webcache

to the browser.

In the event the browser running on an end-user node interfaced with an intermediate PIC component on a remote terminal did not originate the HTTP request that generated an HTTP response provided by the secondary monitoring component at the intermediate node to the buffering component at the intermediate node, and the buffering component at the intermediate node has not yet received via an encapsulated UDP message from the hub node containing a copy of the HTTP request that triggered such HTTP response, the buffering component at the intermediate node stores the HTTP response in buffer memory, tagged by source socket and destination socket, until the correct (matchable sockets) HTTP request arrives via an encapsulated UDP message from the hub node. Upon receipt and decapsulation of the HTTP request, the buffering component can match the HTTP request with the correct HTTP response, extract the URL identity from the HTTP request, and create a file (or database record) for the URL on the storage device(s) used by a PIC Webcache as if the PIC Webcache itself had cached the URL and its contents.

Third Embodiment: Buffering and PIC Webcache at intermediate node (TR: Socket Matching at IN)

FIG. 4 illustrates the Passive Internet Cache embodiment with socket matching and PIC Webcache at an intermediate node in a star topology, dual channel network. FIG. 4A illustrates the distribution of PIC components in the third embodiment. Reference numerals are to FIG. 4 unless otherwise noted.

A third embodiment of the Passive Internet Cache system in a star topology, dual channel network implements a buffering component (406) and a PIC Webcache (407) an intermediate node (417). This embodiment has the advantage of requiring only one instance of the secondary monitoring component, buffering component, and PIC Webcache to serve all end-user nodes on one or more local area networks (418) (or other local connectivity) interfaced with the intermediate node, and providing improved response time without the expense of have end-user nodes equipped with caching devices. Additionally, high performance memory, storage, and dedicated processor(s) can provide performance improvements by off-loading processing and storage requirements from the end-user nodes. Such a configuration is well-suited to a residential set-top box used as an intermediate node, with home PCs, appliances, security, and environmental systems as end-user nodes. TCP/IP is used in the local area network (or other connectivity) between the intermediate node and the end-user nodes.

The third embodiment of the Passive Internet Cache system implements a primary monitoring (401) and an encapsulation component (402) at a hub node, a secondary monitoring component (404), decapsulation component (405), buffering component (406),

and PIC Webcache (407) at a computer or set-top box that is interfaced with, or is an integral part of, a remote terminal (409) and is also interfaced with a local area network (418) (or other connectivity) that serves one or more end-user nodes. The end-user nodes may also communicate directly with a remote terminal for communications unrelated to the PIC system. The PIC Webcache in this embodiment may be configured as a proxy server cache (i.e., browsers configured to use a TCP port for HTTP that is uniquely allocated to the PIC Webcache) or as a caching appliance (i.e., browsers need not be configured to use a TCP port for HTTP that is uniquely allocated to the PIC Webcache, and would normally use well-known TCP port 80). The Web browser on the end-user node must be configured for proxy server cache if the proxy configuration is implemented. If the PIC Webcache is configured as a caching appliance, it is not necessary to configure browsers on end-user nodes to use a caching appliance, since using well-known TCP port 80 in the destination socket directs HTTP requests through the caching appliance. The primary monitoring component (401) and the encapsulation component (402) of the PIC system at the hub node function (403 in FIG. 4A) as described for the first and second embodiments. In addition, the secondary monitoring component (404), and the decapsulation component (405), at the intermediate node communicate and function as described in the second embodiment. The buffering component (406) at the intermediate node communicates with disk drive(s) (412) or other read/write storage device(s) associated with the intermediate node that are used by one or more PIC Webcaches (407) at the intermediate node for storage. The buffering component at the intermediate node stores the decapsulated HTTP request in buffer memory, and opens a file (or database record) tagged with the source socket (source IP address and TCP port number) and the destination socket (destination IP address and TCP port number) specified in the decapsulated HTTP request. Multiple PIC Webcaches may be associated with the intermediate node to enable load-sharing or redundancy. The tagged file (or database record) would normally be on the storage device(s) used by the PIC Webcache at the intermediate node. The tagged file (or database record) awaits the arrival of the contents of the relevant HTTP response to be delivered later by the buffering component.

Upon receipt at the hub node of an HTTP response, the hub node (403 in FIG 4A) transmits the HTTP response in the outbound broadcast channel (416 in FIG. 4). If a browser running on a end-user node interfaced through a PIC-equipped intermediate node to a remote terminal originated the HTTP request that generated the HTTP response, the HTTP response is received in the outbound channel by that remote terminal, forwarded to the PIC Webcache in the normal manner, and delivered by the PIC Webcache, operating either as a proxy server cache or as a caching appliance, to the requesting browser. The secondary monitoring component at the intermediate node forwards through a second

communications path (411) every HTTP response in the outbound channel to the buffering component running on the intermediate node. If a browser running on an end-user node interfaced through a PIC-equipped intermediate node to a remote terminal did not originate the HTTP request that generated the HTTP response, the contents of the HTTP response are not passed to the browser upon receipt. Rather, the buffering component on the intermediate node examines the HTTP response to determine the source and destination sockets and stores the contents of the HTTP response in the file (or database record) previously tagged with those sockets (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response). After receiving both the HTTP request and the HTTP response, the buffering component at the intermediate node can match the HTTP request with the correct HTTP response, extract the URL identity from the HTTP request, and create a file (or database record) for the URL on the storage device(s) used by the PIC Webcache(s) at the intermediate node as if the relevant PIC Webcache itself had cached the URL. To receive and store the HTTP response, the PIC system does not obtain or use a TCP port number allocated by end-user nodes interfaced with the intermediate node. If a browser on an end-user node requests that URL in the future, the contents of the URL will already be in the PIC Webcache, and will be delivered from the PIC Webcache to the browser.

In the event the browser running on an end-user node interfaced with a remote terminal did not originate the HTTP request that generated an HTTP response provided by the secondary monitoring component at the intermediate node to the buffering component at the intermediate node, and the buffering component at the intermediate node has not yet received, via an encapsulated UDP message from the hub node, a copy of the HTTP request that triggered such HTTP response, the buffering component at the intermediate node stores the HTTP response in buffer memory, tagged by source socket and destination socket, until the correct (matchable sockets) HTTP request arrives via an encapsulated UDP message from the hub node. Upon receipt and decapsulation of the HTTP request, the buffering component can match the HTTP request with the correct HTTP response, extract the URL identity from the HTTP request, and create a file (or database record) for the URL on the storage device(s) used by the PIC Webcache(s) on the intermediate node as if the relevant PIC Webcache itself had cached the URL and its contents.

Fourth Embodiment: Split Buffering (hub and end-user nodes) and PIC Webcache at end-user node (TR: Socket Matching at hub node ("HN"))

FIG. 5 illustrates the Passive Internet Cache embodiment with socket matching at the hub node and with secondary buffering and PIC Webcache at an end-user node in a star topology, dual channel network. FIG. 5A illustrates the distribution of PIC components in the fourth embodiment. Reference numerals are to FIG. 5 unless otherwise noted.

A fourth embodiment of the Passive Internet Cache system in a star topology, dual channel network implements a primary monitoring component (501), an encapsulation component (502), and a primary buffering component (519) at a hub node, an initial socket matching function in the primary buffering component (519) at the hub node, and a secondary monitoring component (504), a secondary buffering component (506), a decapsulation component (505), and a PIC Webcache (507) at an end-user node (508 in FIG. 5A). In the context of a star topology, dual channel network with a large number of remote terminals, the use of a primary buffering component at the hub node permits a smaller buffer memory, lower performance storage, and a lower performance processor to be used by the secondary buffering component at the end-user node. Assuming a remote terminal can output all HTTP responses by the use of configuration settings, the typical PIC components at the end user node are software modules installed on an end-user PC and, as such, require no hardware investment to implement. The PIC Webcache is configured as a proxy server cache and the Web browser on the end-user computer is configured to use such proxy server cache. The primary monitoring component (501) at the hub node monitors all inbound channels transmitted from all the remote terminals (509, 513) to the hub node . The primary monitoring component (501) copies all HTTP requests from the monitored inbound channels (514, 515), and forwards each HTTP request to the encapsulation component (502). The encapsulation component creates a UDP packet containing the HTTP request using a broadcast IP address and port as described above, and forwards the encapsulated packet to the primary buffering component (519) at the hub node. The primary buffering component (519) stores the encapsulated HTTP request in buffer memory, tagged with the source socket (source IP address and TCP port number) and the destination socket (destination IP address and TCP port number) specified in the HTTP request. The primary buffering component at the hub node examines the source and destination sockets in the IP packet containing the HTTP response and matches them with the destination and source sockets in the appropriate previously UDP-encapsulated IP packet containing the HTTP request. Upon receipt at the hub node of an HTTP response, the primary monitoring component (501) provides the HTTP response to the primary buffering component (519) at the hub node. The HTTP response is not immediately broadcast by the hub node in the outbound channel. The primary buffering component (519) at the hub node examines the source and destination sockets in the IP packet containing the HTTP response and matches them with the destination and source sockets in the corresponding, previously UDP-encapsulated IP packet containing the HTTP request (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response). The primary buffering component (519), soon after providing the encapsulated HTTP request to the outbound channel (516), provides the corresponding

HTTP response to the outbound channel (516) for transmission to the remote terminals (509, 513).

The secondary monitoring component (504) at the end-user node (508, in FIG. 5A) communicates through a network interface on the end-user node with the remote terminal (509). The secondary monitoring component (504) also communicates through a first communications path (510) with the decapsulation component (505), and thence with the secondary buffering component (506), at the end-user node. The secondary monitoring component (504) copies from the remote terminal through the first communications path (510) to the decapsulation component (505) on the end-user node all encapsulated UDP packets, and through a second communications path (511) to the secondary buffering component on the end-user node all HTTP packets, received by the remote terminal, not just those packets with an IP destination address of the end-user node. When the secondary monitoring component (504) on the end-user node receives a UDP packet containing an encapsulated HTTP request, it provides through a first portion of the first communications path (510) such packet to the decapsulation component. The decapsulation component removes the UDP encapsulation ("decapsulation"), and forwards through a second portion of the first communications path (510) each decapsulated HTTP request to the buffering component at the end-user node. The buffering component at the end-user node communicates with disk drive(s) or other read/write storage device(s) associated with the end user node that are used by the PIC Webcache for storage. The buffering component stores the decapsulated HTTP request in buffer memory, and opens a file (or database record) tagged with the source socket (source IP address and TCP port number) and the destination socket (destination IP address and TCP port number) specified in the decapsulated HTTP request. The tagged file (or database record) would normally be on the storage device(s) used by the PIC Webcache. The tagged file (or database record) awaits the arrival of the contents of the relevant HTTP response from the buffering component.

If a browser running on a end-user node interfaced with a remote terminal originated the HTTP request that generated the HTTP response, the HTTP response is received in the outbound channel by that remote terminal, forwarded to the PIC Webcache in the normal manner, and delivered by the PIC Webcache, operating as a proxy server cache, to the requesting browser. The secondary monitoring component (504) at the end-user node also forwards through the second communications path (511) every HTTP response in the outbound channel to the secondary buffering component (506) running on the end-user node. If a browser running on an end-user node interfaced with a remote terminal did not originate the HTTP request that generated the HTTP response, the contents of the HTTP response are not passed to the browser upon receipt. Rather, the

secondary buffering component on the end-user node examines the HTTP response to determine the source and destination sockets and stores the contents of the HTTP response in the file (or database record) previously tagged with those sockets (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response). After receiving both the HTTP request and the HTTP response, the secondary buffering component at the end-user node can match the HTTP request with the correct HTTP response, extract the URL identity from the HTTP request, and create a file (or database record) for the URL on the storage device(s) (512) used by the PIC Webcache (507) as if the PIC Webcache itself had cached the URL. To receive and store the HTTP response, the PIC system does not obtain or use a TCP port number allocated by the end-user node. If the browser requests that URL in the future, the contents of the URL will already be in the PIC Webcache serving the browser, and will be delivered instantly from the PIC Webcache to the browser.

Fifth Embodiment: Split Buffering (hub node and intermediate node) and PIC Webcache at end-user node (TR: Socket Matching at HN)

FIG. 6 illustrates the Passive Internet Cache embodiment with socket matching at the hub node, with secondary buffering at an intermediate node, and with PIC Webcache at an end-user node in a star topology, dual channel network. FIG. 6A illustrates the distribution of PIC components in the fifth embodiment. Reference numerals are to FIG. 6 unless otherwise noted.

A fifth embodiment of the Passive Internet Cache system in a star topology, dual channel network implements a primary monitoring component (601), an encapsulation component (602), and a primary buffering component (619) at the hub node, an initial socket matching function in the primary buffering component at the hub node, and a secondary monitoring component (604), a secondary buffering component (606), and a decapsulation component (605) at an intermediate node, and a PIC Webcache (607) at an end-user node. In the context of a star topology, dual channel network with a large number of remote terminals, the use of a primary buffering component at the hub node permits smaller buffer memory and lower performance storage and processor to be used by the secondary buffering component at the intermediate node. The PIC Webcache is configured as a proxy server cache and the Web browser on the end-user computer is configured to use such proxy server cache. The primary monitoring component at the hub node monitors all inbound channels transmitted from all the remote terminals to the hub node. The primary monitoring component (601) copies all HTTP requests from the monitored inbound channels, and provides each HTTP request to the encapsulation component (602) where, as in the fourth embodiment, each HTTP request is encapsulated in a UDP packet with a broadcast IP address and port and is then forwarded to the primary

buffering component (619) at the hub node. The primary buffering component (619) stores the encapsulated HTTP request in buffer memory at the hub node, tagged with the source socket (source IP address and TCP port number) and the destination socket (destination IP address and TCP port number) specified in the HTTP request. Upon receipt at the hub node of an HTTP response, the primary monitoring component (601) provides the HTTP response to the primary buffering component (619) at the hub node. The HTTP response is not immediately broadcast by the hub node in the outbound channel. The primary buffering component (619) at the hub node examines the source and destination sockets in the IP packet containing the HTTP response and matches them with the destination and source sockets in the associated, previously UDP-encapsulated IP packet containing the HTTP request (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response). After receiving both the HTTP request and the HTTP response, the primary buffering component (619) provides the HTTP request to the outbound channel (616) for transmission in the outbound channel using a broadcast IP address and UDP port (or other suitable protocol, address, and port) monitored by all remote terminals that receive such outbound channel and are interfaced with local PIC components. The primary buffering component (619), soon after providing the encapsulated HTTP request to the outbound channel (616), provides the corresponding HTTP response to the outbound channel (616) for transmission to the remote terminals (609, 613).

The secondary monitoring component (604) at the intermediate node (617 in FIG. 6A) communicates through a network interface on the intermediate node with the remote terminal. The secondary monitoring component (604) also communicates with the decapsulation component (605), and with the secondary buffering component (606), at the intermediate node. The secondary monitoring component (604) copies from the remote terminal (603) and forwards through a first portion of a first communications path (610) to the decapsulation component (605) on the intermediate node all encapsulated UDP packets, and through a second communications path (611) to the secondary buffering component on the intermediate node all HTTP packets, received by the remote terminal, not just those packets with an IP destination address of the intermediate node or of end-user nodes served by that remote terminal. When the secondary monitoring component (604) on the intermediate node receives a UDP packet containing an encapsulated HTTP request, it provides such packet to the decapsulation component through the first portion of the first communications path (610). The decapsulation component removes the UDP encapsulation ("decapsulation"), and forwards through a second portion of the first communications path (611) each decapsulated HTTP request to the secondary buffering component (606) at the intermediate node. The secondary buffering component (606) at

the intermediate node communicates with disk drive(s) (612) or other read/write storage device(s) associated with each end user node that are used by one or more end-user nodes' PIC Webcaches for storage. The secondary buffering component stores the decapsulated HTTP request in buffer memory, and opens a file (or database record) tagged with the source socket (source IP address and TCP port number) and the destination socket (destination IP address and TCP port number) specified in the decapsulated HTTP request. The tagged file (or database record) would normally be on the storage device(s) used by each PIC Webcache interfaced with the intermediate node. The tagged file (or database record) awaits the arrival of the contents of the relevant HTTP response from the buffering component.

If a browser running on a end-user node interfaced with a remote terminal originated the HTTP request that generated the HTTP response, the HTTP response is received in the outbound channel by that remote terminal, forwarded to each PIC Webcache in the normal manner, and delivered by the PIC Webcache associated with an end-user node, operating as a proxy server cache, to the requesting browser. The secondary monitoring component (604) at the intermediate node also forwards through the second communications path (611) every HTTP response in the outbound channel to the secondary buffering component (606) running on the intermediate node. If a browser running on an end-user node interfaced with a remote terminal did not originate the HTTP request that generated the HTTP response, the contents of the HTTP response are not passed to the browser upon receipt. Rather, the secondary buffering component on the intermediate node examines the HTTP response to determine the source and destination sockets and stores the contents of the HTTP response in the file (or database record) previously tagged with those sockets (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response). After receiving both the HTTP request and the HTTP response, the secondary buffering component at the intermediate node can match the HTTP request with the correct HTTP response, extract the URL identity from the HTTP request, and create a file (or database record) for the URL on the storage device(s) used by each PIC Webcache as if each PIC Webcache itself had cached the URL. To receive and store the HTTP response, the PIC system typically uses a local area network (618) and may use a TCP port number allocated by an end-user node for the purpose of directly accessing the disk area of the PIC Webcache on the end-user node. If a browser requests that URL in the future, the contents of the URL will already be in the PIC Webcache serving the browser, and will be delivered instantly from the PIC Webcache to the browser.

Sixth Embodiment: Split Buffering (hub node and intermediate node) and PIC Webcache at intermediate node (TR: Socket Matching at HN)

FIG. 7 illustrates the Passive Internet Cache embodiment with socket matching at the hub node and with PIC Webcache at an intermediate node in a star topology, dual channel network. FIG. 7A illustrates the distribution of PIC components in the sixth embodiment. Reference numerals are to FIG. 7 unless otherwise noted.

A sixth embodiment of the Passive Internet Cache system in a star topology, dual channel network implements a primary monitoring component (701), an encapsulation component (702), and a primary buffering component (719) at a hub node, an initial socket matching function in the primary buffering component (719) at the hub node, and a secondary monitoring component (704), a secondary buffering component (706), a decapsulation component (705), and a PIC Webcache (707) at an intermediate node (717 in FIG. 7A). In the context of a star topology, dual channel network with a large number of remote terminals, the use of a primary buffering component at the hub node permits smaller buffer memory and lower performance storage and processor to be used by the secondary buffering component at the intermediate node. This embodiment has the advantage of requiring only one instance of the secondary monitoring component, buffering component, and PIC Webcache to serve all end-user nodes on one or more local area networks (or other local connectivity) interfaced with the intermediate node, and providing improved response time without the expense of have end-user nodes equipped with caching devices. Additionally, high performance memory, storage, and dedicated processor can provide performance improvements by off-loading processing and storage requirements from the end-user nodes. The end-user nodes may also communicate directly with a remote terminal for communications unrelated to the PIC system. The PIC Webcache in this embodiment may be configured as a proxy server cache (i.e., browsers configured to use a TCP port for HTTP that is uniquely allocated to the PIC Webcache) or as a caching appliance (i.e., browsers need not be configured to use a TCP port for HTTP that is uniquely allocated to the PIC Webcache, and would normally use well-known TCP port 80). The Web browser on the end-user node must be configured for proxy server cache if the proxy configuration is implemented. If the PIC Webcache is configured as a caching appliance, it is not necessary to configure browsers on end-user nodes to use the caching appliance, since using well-known TCP port 80 in the destination socket directs HTTP requests through the caching appliance.

The primary monitoring component (701), the primary buffering component (719), and the encapsulation component (702) of the PIC system at the hub node function as described above for the fourth and fifth embodiments. In addition, the secondary monitoring component (704), the decapsulation component (705), first communications path (710), and second communications path (711) at the intermediate node function as described above for the fifth embodiment. In the sixth embodiment, the secondary

buffering component (706) at the intermediate node communicates with disk drive(s) (712) or other read/write storage device(s) located at the intermediate node that are used by one or more PIC Webcaches at the intermediate node for storage. The secondary buffering component (706) at the intermediate node stores the decapsulated HTTP request in buffer memory, and opens a file (or database record) tagged with the source socket (source IP address and TCP port number) and the destination socket (destination IP address and TCP port number) specified in the decapsulated HTTP request. The tagged file (or database record) would normally be on the storage device(s) used by a given PIC Webcache at the intermediate node. Multiple PIC Webcaches may be associated with the intermediate node to enable load-sharing or redundancy. The tagged file awaits the arrival of the contents of the relevant HTTP response from the buffering component.

If a browser running on a end-user node interfaced with a remote terminal originated the HTTP request that generated the HTTP response, the HTTP response is received in the outbound channel by that remote terminal, forwarded to the relevant PIC Webcache in the normal manner, and delivered by the PIC Webcache, operating either as a proxy server cache or as a cascaded caching appliance, to the requesting browser. The secondary monitoring component at the intermediate node forwards every HTTP response in the outbound channel to the secondary buffering component running on the intermediate node. If a browser running on an end-user node interfaced with a remote terminal did not originate the HTTP request that generated the HTTP response, the contents of the HTTP response are not passed to the browser upon receipt. Rather, the secondary buffering component on the intermediate node examines the HTTP response to determine the source and destination sockets and stores the contents of the HTTP response in the file (or database record) previously tagged with those sockets (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response). After receiving both the HTTP request and the HTTP response, the secondary buffering component at the intermediate node can match the HTTP request with the correct HTTP response, extract the URL identity from the HTTP request, and create a file (or database record) for the URL on the storage device(s) used by one or more PIC Webcaches at the intermediate node as if the PIC Webcache itself had cached the URL and its contents. To receive and store the HTTP response, the PIC system does not obtain or use a TCP port number allocated by end-user nodes interfaced with the intermediate node. If a browser on an end-user node requests that URL in the future, the contents of the URL will already be in the relevant PIC Webcache, and will be delivered from the PIC Webcache to the browser.

Seventh Embodiment: Buffering and PIC Webcache at end-user node (RO: Socket Matching at EUN)

FIG. 8 illustrates the Passive Internet Cache embodiment with socket matching and

PIC Webcache at an end-user node in a star topology, single channel network. FIG. 8A illustrates the distribution of PIC components in the seventh embodiment. Reference numerals are to FIG. 8 unless otherwise noted.

A seventh embodiment of the PIC system is structurally and functionally identical the first embodiment, except the remote terminal is receive only and cannot initiate HTTP requests. The inability to initiate HTTP requests, or other return channel messages, does not prevent or impair the operation of the components of the PIC system. The PIC Webcache built at an end-user node reflects the HTTP requests initiated by other end-user nodes that have an inbound channel.

Eighth Embodiment: Buffering at intermediate node and PIC Webcache at end-user node (RO: Socket Matching at IN)

FIG. 9 illustrates the Passive Internet Cache embodiment with socket matching at an intermediate node and with PIC Webcache at an end-user node in a star topology, single channel network. FIG. 9A illustrates the distribution of PIC components in the eighth embodiment. Reference numerals are to FIG. 9 unless otherwise noted.

An eighth embodiment of the PIC system is structurally and functionally identical the second embodiment, except the remote terminal is receive only and cannot initiate HTTP requests. The inability to initiate HTTP requests, or other return channel messages, does not prevent or impair the operation of the components of the PIC system. The PIC Webcache built at an end-user node reflects the HTTP requests initiated by other end-user nodes that have an inbound channel.

Ninth Embodiment: Buffering and PIC Webcache at intermediate node (RO: Socket Matching at IN)

FIG. 10 illustrates the Passive Internet Cache embodiment with socket matching and PIC Webcache at an intermediate node in a star topology, single channel network. FIG. 10A illustrates the distribution of PIC components in the ninth embodiment. Reference numerals are to FIG. 10 unless otherwise noted.

A ninth embodiment of the PIC system is structurally and functionally identical the third embodiment, except the remote terminal is receive only and cannot initiate HTTP requests. The inability to initiate HTTP requests, or other return channel messages, does not prevent or impair the operation of the components of the PIC system. The PIC Webcache built at an intermediate node reflects the HTTP requests initiated by other end-user nodes that have an inbound channel.

Tenth Embodiment: Split Buffering (hub and end-user nodes) and PIC Webcache at end-user node (RO: Socket Matching at HN)

FIG. 11 illustrates the Passive Internet Cache embodiment with socket matching at the hub node and with secondary buffering and PIC Webcache at an end-user node in

a star topology, single channel network. FIG. 11A illustrates the distribution of PIC components in the tenth embodiment. Reference numerals are to FIG. 11 unless otherwise noted.

A tenth embodiment of the PIC system is structurally and functionally identical the fourth embodiment, except the remote terminal is receive only and cannot initiate HTTP requests. The inability to initiate HTTP requests, or other return channel messages, does not prevent or impair the operation of the components of the PIC system. The PIC Webcache built at an end-user node reflects the HTTP requests initiated by other end-user nodes that have an inbound channel.

Eleventh Embodiment: Split Buffering (hub node and intermediate node) and PIC Webcache at end-user node (RO: Socket Matching at HN)

FIG. 12 illustrates the Passive Internet Cache embodiment with socket matching at the hub node, with secondary buffering at an intermediate node, and with PIC Webcache at an end-user node in a star topology, single channel network. FIG. 12A illustrates the distribution of PIC components in the eleventh embodiment. Reference numerals are to FIG. 12 unless otherwise noted.

An eleventh embodiment of the PIC system is structurally and functionally identical the fifth embodiment, except the remote terminal is receive only and cannot initiate HTTP requests. The inability to initiate HTTP requests, or other return channel messages, does not prevent or impair the operation of the components of the PIC system. The PIC Webcache built at an end-user node reflects the HTTP requests initiated by other end-user nodes that have an inbound channel.

Twelfth Embodiment: Split Buffering (hub node and intermediate node) and PIC Webcache at intermediate node (RO: Socket Matching at HN)

FIG. 13 illustrates the Passive Internet Cache embodiment with socket matching at the hub node and with secondary buffering and PIC Webcache and at an intermediate node in a star topology, single channel network. FIG. 13A illustrates the distribution of PIC components in the twelfth embodiment. Reference numerals are to FIG. 13 unless otherwise noted.

A twelfth embodiment of the PIC system is structurally and functionally identical the sixth embodiment, except the remote terminal is receive only and cannot initiate HTTP requests. The inability to initiate HTTP requests, or other return channel messages, does not prevent or impair the operation of the components of the PIC system. The PIC Webcache built at an intermediate node reflects the HTTP requests initiated by other end-user nodes that have an inbound channel.

For simplicity, in the embodiments that do not allocate TCP ports, if a browser running on an end-user node interfaced with a remote terminal originated the HTTP

request that generated a given HTTP response, a PIC system would normally be configured so that the same HTTP request after being decapsulated from a UDP message, after receipt of the socket-matched HTTP response, causes an overwrite of the URL in the PIC Webcache that was already written in response to the HTTP response addressed directly to that end-user node's IP address and TCP port number (rather than through the PIC system). Such an implementation is said to have "exhaustive writing" of HTTP requests received by the PIC buffering component through UDP messages (i.e., does not need to recognize local addresses before writing the URL processed through the PIC system). Alternatively, a PIC buffering component on an end-user node can be designed for "selective writing" so that it does not write any HTTP request received by the PIC monitoring component through UDP messages to the PIC Webcache if the destination IP address of the HTTP response is that of the node on which the PIC Webcache is running. In case of "selective writing", the normal dialog of an HTTP request and an HTTP response will cause the requested URL to be written to the browser or cache that originated the HTTP request.

If the necessary decryption keys are available, a PIC system can work with encrypted HTTP messages. If encrypted HTTP messages are exchanged, and the necessary decryption key is not available to a PIC system, the PIC system is unable to use (or possibly detect) HTTP messages that are encrypted. In all embodiments of the PIC system, encrypted payloads are simply handled as any other message, i.e. cached in the encrypted form if TCP/IP header information is available and HTTP commands are decipherable, or discarded if not. Independently of the PIC system components, encrypted HTTP messages are processed in a normal manner, so that a browser sending an encrypted HTTP request can still receive an encrypted HTTP response.

The return channel from a hub node through one or more monitoring components, encapsulating component, decapsulating component, one or more buffering components, to each PIC Webcache can also carry UDP messages that can be used to control HTTP content cached by a PIC Webcache based on the URL in an HTTP request or the source IP address of an HTTP response. Such access control can be used to permit or prevent caching of URLs and their content in a PIC Webcache in order to implement a subscription authorization system, or simply to block content from objectionable network domains, hosts, etc. With reference to FIG. 16, implementation of access control using a PIC system requires a content authorization program and database associated with or interfaced with the hub node (1620), preparation of content authorization messages addressed to one or more buffering components associated with remote terminals by the content authorization program, encapsulation of the authorization control messages by the encapsulating component (1602) at a hub node, transmission of the encapsulated

authorization control messages in the outbound channel (1616), delivery of the decapsulated authorization control messages to the each buffering component in the same manner as delivery of decapsulated HTTP requests as described above, and execution of the authorization control message by an intermediate node content authorization component (1621) associated with a buffering component (1606) and/or by an end-user node content authorization component (1622) associated with a PIC Webcache (1607). The authorization control message determines what URLs are cached in a given PIC Webcaches; messages exchanged between the buffering component (1606) and a given intermediate node or end-user node content authorization component determine what URLs are blocked from caching. Content authorization components (1621, 1622) associated with or interfaced with a remote terminal prevent a PIC Webcache from caching a URL or source IP address that is specified as unauthorized ("blocked") for a given PIC Webcache. Content authorization through a content authorization program and database is known in the art, but its use in conjunction with a PIC system is novel. A content authorization component at different end-user nodes can be used to create end-user PIC Webcaches with different content, but associated with a single intermediate node (1617) and intermediate node content authorization component (1621). By using a content authorization component (1621) at an intermediate node (1617), an additional point of content control is introduced. Content authorization at an intermediate node PIC Webcache (not illustrated) can be implemented in networks that have some end-user nodes equipped with PIC Webcaches and some that are not equipped with PIC Webcaches. Different end-user nodes can also be assigned to use different PIC Webcaches, and thereby have access to different content.

The buffering component at an intermediate node or at an end-user node can contain a program that monitors whether the payload of an HTTP request or HTTP response has errors based on common error checking methods, such as packet sequence number continuity. A given HTTP message may require multiple TCP/IP packets for transmission. With reference to FIG. 17, TCP inserts a sequence number into each packet used to transmit a given HTTP message. For a given HTTP response provided to a PIC buffering component associated with or interfaced with a remote terminal, only one payload of a packet with a given TCP sequence number is needed. If a packet stream for a given HTTP request contains more than one packet with the same TCP sequence number (upper portion of FIG. 17), the buffering component discards packets with duplicate TCP sequence numbers. Such error checking is without the buffering component's participation in normal TCP/IP protocol handshakes. By reference to the lower portion of FIG. 17, a PIC buffering component associated with or interfaced with a remote terminal can be configured to discard HTTP messages determined by the buffering component to have

missing TCP sequence numbers. If the remote terminal associated with or interfaced with a buffering component has inbound channel capability, and the buffering component determines that an HTTP response has a corrupted payload, the buffering component can (optionally) use the URL in the decapsulated HTTP request to generate an HTTP request from the end-user node (using normal TCP/IP procedures) for the URL at issue, and obtain another HTTP response from the relevant Webserver.

Taking an "end to end" perspective of the handling of UDP messages and HTTP responses by PIC components, in PIC embodiments without socket-matching at a hub node, for UDP messages, a first end-to-end communications path exists through a primary monitoring component and encapsulating component at a hub node, then through an outbound channel from the hub node, then through a remote terminal to a secondary monitoring component and decapsulating component to a buffering component; for HTTP responses, a second end-to-end communications path exists from external server computers or traditional caches through the outbound channel of the hub node, then through the remote terminal to the secondary monitoring component, and then to the buffering component. In PIC embodiments with socket-matching at a hub node, for UDP messages, a first end-to-end communications path exists through a primary monitoring component, encapsulating component, and primary buffering component at a hub node, through the outbound channel from the hub node, then through a remote terminal to a secondary monitoring component and decapsulating component, then to a secondary buffering component; for HTTP responses, a second end-to-end communications path exists from external server computers or traditional caches through the primary buffering component at a hub node, then through the outbound channel from the hub node, then through a remote terminal to the secondary monitoring component and then to the secondary buffering component.

Those skilled in the art also will readily appreciate that many modifications to the invention are possible within the scope of the invention. Another alternative embodiment, to provide redundancy, would include more than one PIC buffering component associated with a given end-user node. For mixed populations of end-user nodes, an intermediate node PIC Webcache can be implemented in networks that have some end-user nodes equipped with PIC Webcaches and some that are not equipped with PIC Webcaches. Accordingly, the scope of the invention is not intended to be limited to the preferred embodiments described above, but only by the appended claims.

CLAIMS

We claim:

1. A system of encapsulation and passive listening, comprising:

a means hosted on a hub node of a network by which a message from a first network node to said hub node is encapsulated in a broadcast or multicast message and transmitted to one or more other network nodes unable to receive the original transmission of said message from said first network node to said hub node,

a means associated with each of said one or more other network nodes of receiving, unencapsulating, and buffering said message when it arrives, and

a means of storing the contents of the decapsulated and buffered message in cache memory or storage accessible by an end-user at each of said one or more other network nodes.

2. A system of encapsulation and passive listening, comprising:

a means hosted on a hub node of a network by which a message from a first network node to said hub node is selected for encapsulation based on one or more criteria, such as being an HTTP request or being addressed to a certain network address or socket, then encapsulated in a broadcast or multicast message and transmitted to one or more other network nodes unable to receive the original transmission of said message from said first network node to said hub node,

a means associated with each of said one or more other network nodes of receiving, unencapsulating, and buffering said message when it arrives, and

a means of storing the contents of the decapsulated and buffered message in cache memory or storage accessible by an end-user at each of said one or more other network nodes.

3. A system of encapsulation and passive listening, according to claim 1 or 2, wherein said message transmitted from said hub node to said one or more other network nodes is encapsulated in User Datagram Protocol or a transport layer, connectionless protocol substantially equivalent to User Datagram Protocol.

4. A system of socket matching, comprising:

a means hosted on a hub node of a network by which an HTTP request from a first network node to said hub node is forwarded to a server computer and by which means a copy of said HTTP request is also transmitted to one or more other network nodes unable to receive the original transmission of said HTTP request from said first network node to said hub node, and by which means a copy of an HTTP response sent by said server computer in response to said HTTP request is transmitted to said first network node and to said one or more other network nodes,

a means, associated with or interfaced with each of said one or more other network

nodes, of receiving and buffering said HTTP request when it arrives, and of receiving and buffering said HTTP response when it arrives, in buffer memory associated with or interfaced with each of said one or more other network nodes,

a means, associated with or interfaced with each of said one or more other network nodes, of matching the source socket and destination socket of a buffered HTTP request with the source socket and destination socket of a buffered HTTP response (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response) sent by the server computer in response to said HTTP request and received before or after receipt of said HTTP request, and

a means of storing under the URL specified in the matching HTTP request the contents of the matched HTTP response in cache memory accessible to an end-user node, which end user node is associated with or interfaced with one of said one or more other network nodes.

5. A system of socket matching, comprising:

a means hosted on a hub node of a network by which an HTTP request from a first network node to said hub node is forwarded to a server computer and by which means a copy of said HTTP request is buffered at said hub node until such time as an HTTP response sent by said server computer in response to said HTTP request is received, buffered, and matched as described below at said hub node,

a means hosted on said hub node of matching the source socket and destination socket of said buffered HTTP request with the source socket and destination socket of said buffered HTTP response (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response),

a means hosted on said hub node of transmitting a matched HTTP response and the matching HTTP request to one or more other network nodes unable to receive the original transmission of said HTTP request from said first network node to said hub node,

a means, associated with or interfaced with each of said one or more other network nodes, of receiving and buffering said HTTP request when it arrives, and of receiving and buffering said HTTP response when it arrives, in buffer memory associated with or interfaced with each of said one or more other network nodes,

a means, associated with or interfaced with each of said one or more other network nodes, of matching the source socket and destination socket of a buffered HTTP request with the source socket and destination socket of a buffered HTTP response (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response) sent by the server computer in response to said HTTP request and received before or after receipt of said HTTP request, and

a means of storing under the URL specified in said matching HTTP request the

contents of said matched HTTP response in cache memory accessible to an end-user node, which end user node is associated with or interfaced with one of said one or more other network nodes.

6. A system of socket matching, according to claim 4 or 5, wherein Transmission Control Protocol or a transport layer protocol other than Transmission Control Protocol that uses the network addressing and routing capabilities of Internet Protocol, or a protocol other than Hypertext Transport Protocol that uses paired network layer addressing and transport layer ports in request/response, client/server exchanges substantially equivalent to such use in Hypertext Transport Protocol, is used.

7. A system of socket matching, according to claim 4 or 5, wherein the network is a star topology network selected from the group consisting of very small aperture satellite, multichannel multipoint distribution service, local multipoint distribution service, instructional television fixed service, digital television broadcasting, and digital cablecasting.

8. A system of socket matching, according to claim 4 or 5, wherein each said copy of an HTTP request is encapsulated in a broadcast or multicast message before transmission from said hub node and is decapsulated upon receipt and prior to buffering at each of said one of more other network nodes.

9. A caching system for a star topology network, comprising:

a hub node of the star topology network that receives HTTP requests from one or more remote terminals, which hub node is interfaced with at least one external server computer and transmits UDP datagrams, HTTP requests, and HTTP responses in an outbound channel from said hub node of the star topology network, and

one or more intermediate or end-user nodes of said network that receive said UDP datagrams, HTTP requests, and HTTP responses and which one or more intermediate or end-user nodes are associated with or interfaced with a means for socket-matching HTTP requests and HTTP responses and for storing in an end-user accessible cache the contents of a given HTTP response under the URL extracted from a correctly matched HTTP request.

10. A caching system for a star topology network, comprising:

a hub node of the star topology network that receives HTTP requests from one or more remote terminals, which hub node is interfaced with at least one external server computer, has a first means for socket-matching, and transmits UDP datagrams, HTTP requests, and HTTP responses in an outbound channel from said hub node of the star topology network, and

one or more intermediate or end-user nodes of said network that receive said UDP datagrams, HTTP requests, and HTTP responses and which one or more intermediate or

end-user nodes are associated with or interfaced with a second means for socket-matching HTTP requests and HTTP responses and for storing in an end-user accessible cache the contents of a given HTTP response under the URL extracted from a correctly matched HTTP request.

11. A caching system for a star topology network, comprising:

a hub node of the star topology network that has access to one or more external server computers, and internal to the star topology network has associated with the hub node a primary monitoring component, an encapsulating component, and an outbound channel,

a secondary monitoring component, a decapsulating component, and a socket-matching buffering component associated with one or more intermediate node or end-user nodes interfaced with a remote terminal in the star topology network,

at least one global cache associated with said buffering component and an end-user node, or where said buffering component is associated with an intermediate node, said at least one global cache either associated with said buffering component, or associated with an end-user node and interfaced with said buffering component,

communications paths for UDP and HTTP messages from the hub node through the outbound channel to each buffering component,

a means controlled by each associated or interfaced buffering component of writing files or database records in each associated or interfaced global cache,

wherein each global cache is accessible by at least one end-user node and contains information or data from said server computers, and

at least one end-user node is able to exchange HTTP requests and HTTP responses with said hub node using the star network medium or a different telecommunications medium.

12. A caching system for a star topology network, comprising:

a hub node of the star topology network that has access to one or more external server computers, and internal to the star topology network has associated with the hub node a primary monitoring component, an encapsulating component, a socket-matching primary buffering component, and an outbound channel,

a secondary monitoring component, a decapsulating component, and a socket-matching secondary buffering component associated with one or more intermediate nodes or end-user nodes interfaced with a remote terminal in the star topology network,

at least one global cache associated with said secondary buffering component and an end-user node, or where said secondary buffering component is associated with an intermediate node, said at least one global cache is either associated with said secondary buffering component, or associated with an end-user node and interfaced with said secondary buffering component,

communications paths for UDP and HTTP messages from the hub node through the outbound channel to each secondary buffering component,

a means controlled by each associated or interfaced secondary buffering component of writing files or database records in each associated or interfaced global cache,

wherein each global cache is accessible by at least one end-user node and contains information or data from said server computers, and

at least one end-user node is able to exchange HTTP requests and HTTP responses with said hub node using the star network medium or a different telecommunications medium.

13. A caching system for a star topology network, according to claim 9, 10, 11, or 12, wherein Transmission Control Protocol or a transport layer protocol other than Transmission Control Protocol that uses network addressing and routing capabilities of Internet Protocol is used, or a protocol other than Hypertext Transport Protocol that uses paired network layer addressing and transport layer ports in request/response, client/server exchanges substantially equivalent to such use in Hypertext Transport Protocol is used, or a connectionless protocol other than User Datagram Protocol with broadcast or multicast addressing substantially equivalent to that of User Datagram Protocol is used.

14. A caching system for a star topology network, comprising:

a hub node of the star topology network that has access to one or more external server computers, and internal to the star topology network has associated with the hub node a primary monitoring component, an encapsulating component, and an outbound channel,

at least one end-user node interfaced with a remote terminal in the star topology network, and associated with each such end-user node a secondary monitoring component, a decapsulating component, a socket-matching buffering component, and at least one global cache,

a first communications path from the hub node through the primary monitoring component, encapsulating component, and outbound channel, thence through the remote terminal, the secondary monitoring component and decapsulating component to each said buffering component,

a second communications path from said one or more external server computers through the outbound channel of said hub node, thence through the remote terminal, the secondary monitoring component to each said buffering component,

a means controlled by each said buffering component of writing files or database records in said associated global cache,

wherein said global cache is accessible by said end-user node and contains information or data from said server computers, and

at least one end-user node is able to exchange messages with said hub node using the star network medium or a different telecommunications medium.

15. A caching system for a star topology network, comprising:

a hub node of the star topology network that has access to one or more external server computers, and internal to the star topology network has associated with the hub node a primary monitoring component, an encapsulating component, and an outbound channel,

at least one intermediate node interfaced with a remote terminal in the star topology network, and associated with each such intermediate node a secondary monitoring component, a decapsulating component, and a socket-matching buffering component,

at least one global cache associated with an end-user node and interfaced with said buffering component,

a first communications path from the hub node through the primary monitoring component, encapsulating component, and outbound channel, thence through the remote terminal, the secondary monitoring component and decapsulating component to each said buffering component,

a second communications path from said one or more external server computers through the outbound channel of said hub node, thence through the remote terminal, the secondary monitoring component to each said buffering component,

a means controlled by said buffering component of writing files or database records in said interfaced global cache,

wherein said global cache is accessible by said end-user node and contains information or data from said server computers, and

at least one end-user node is able to exchange messages with said hub node using the star network medium or a different telecommunications medium.

16. A caching system for a star topology network, comprising:

a hub node of the star topology network that has access to one or more external server computers, and internal to the star topology network has associated with the hub node a primary monitoring component, an encapsulating component, and an outbound channel,

at least one intermediate node interfaced with a remote terminal in the star topology network and with at least one end-user node, and associated with each such intermediate node is a secondary monitoring component, a decapsulating component, a socket-matching buffering component, and at least one global cache,

a first communications path from the hub node through the primary monitoring component, encapsulating component, and outbound channel, thence through the remote terminal, the secondary monitoring component and decapsulating component to each said

buffering component,

a second communications path from said one or more external server computers through the outbound channel of said hub node, thence through the remote terminal, the secondary monitoring component to each said buffering component,

a means controlled by each said buffering component of writing files or database records in said associated global cache,

wherein each global cache is accessible by at least one end-user node interfaced with said intermediate node and contains information or data from said server computers, and

at least one end-user node is able to exchange messages with said hub node using the star network medium or a different telecommunications medium.

17. A caching system for a star topology network, comprising:

a hub node of the star topology network that has access to one or more external server computers, and internal to the star topology network has associated with the hub node a primary monitoring component, an encapsulating component, a socket-matching primary buffering component, and an outbound channel,

at least one end-user node interfaced with a remote terminal in the star topology network, and associated with each such end-user node a secondary monitoring component, a decapsulating component, a socket-matching secondary buffering component, and at least one global cache,

a first communications path from the hub node through said primary monitoring component, encapsulating component, primary buffering component, and outbound channel, thence through the remote terminal, said secondary monitoring component and decapsulating component to each said secondary buffering component,

a second communications path from said one or more external server computers through the primary monitoring component, primary buffering component, and outbound channel of said hub node, thence through the remote terminal, said secondary monitoring component to each said secondary buffering component,

a means controlled by each secondary buffering component of writing files or database records in said associated global cache,

wherein said global cache is accessible by said end-user node and contains information or data from said server computers, and

at least one end-user node is able to exchange messages with said hub node using the star network medium or a different telecommunications medium.

18. A caching system for a star topology network, comprising:

a hub node of the star topology network that has access to one or more external

server computers, and internal to the star topology network has associated with the hub node a primary monitoring component, an encapsulating component, a socket-matching primary buffering component, and an outbound channel,

at least one intermediate node interfaced with a remote terminal in the star topology network and interfaced with at least one end-user node, and associated with each such intermediate node a secondary monitoring component, a decapsulating component, and a socket-matching secondary buffering component,

at least one global cache associated with each said end-user node and interfaced with said secondary buffering component,

a first communications path from the hub node through said primary monitoring component, encapsulating component, primary buffering component, and outbound channel, thence through the remote terminal, said secondary monitoring component and decapsulating component to each said secondary buffering component,

a second communications path from said one or more external server computers through the primary monitoring component, primary buffering component, and outbound channel of said hub node, thence through the remote terminal, said secondary monitoring component to each said secondary buffering component,

a means controlled by each secondary buffering component of writing files or database records in said interfaced global cache,

wherein said global cache is accessible by said end-user node and contains information or data from said server computers, and

at least one end-user node is able to exchange messages with said hub node using the star network medium or a different telecommunications medium.

19. A caching system for a star topology network, comprising:

a hub node of the star topology network that has access to one or more external server computers, and internal to the star topology network has associated with the hub node a primary monitoring component, an encapsulating component, a socket-matching primary buffering component, and an outbound channel,

at least one intermediate node interfaced with a remote terminal in the star topology network, and associated with each such intermediate node a secondary monitoring component, a decapsulating component, a socket-matching secondary buffering component, and at least one global cache,

a first communications path from the hub node through said primary monitoring component, encapsulating component, primary buffering component, and outbound channel, thence through the remote terminal, said secondary monitoring component and decapsulating component to each said secondary buffering component,

a second communications path from said one or more external server computers

through the primary monitoring component, primary buffering component, and outbound channel of said hub node, thence through the remote terminal, said secondary monitoring component to each said secondary buffering component,

a means controlled by each said secondary buffering component of writing files or database records in said associated global cache,

wherein each global cache is accessible by at least one end-user node interfaced with said intermediate node and contains information or data from said server computers, and

at least one end-user node is able to exchange messages with said hub node using the star network medium or a different telecommunications medium.

20. A caching system for a star topology network, according to claim 9, 10, 11, 12, 14, 15, 16, 17, 18, or 19, wherein the star networking technology is very small aperture satellite ("VSAT"), multichannel multipoint distribution service ("MMDS"), local multipoint distribution service ("LMDS"), instructional television fixed service ("ITFS"), digital television broadcasting, or digital cablecasting.

21. A caching system for a star topology network, according to claim 9, 10, 11, 12, 14, 15, 16, 17, 18, or 19, wherein a Passive Internet Cache system concurrently uses one or more wireline outbound channels, wireless outbound channels, wireline inbound channels, and/or wireless inbound channels.

22. A caching system for a star topology network, according to claim 14, 15, 16, 17, 18, or 19, wherein the network layer protocol is Internet Protocol or a protocol that uses network addressing and routing substantially equivalent to that of Internet Protocol is used, the transport layer protocol used in the first communications path to encapsulate an HTTP request is User Datagram Protocol or a connectionless protocol substantially equivalent to that of User Datagram Protocol is used, the transport layer protocol used in the second communications path is Transport Control Protocol or a connection-oriented protocol substantially equivalent to Transport Control Protocol is used, or the applications and services protocol is a protocol other than Hypertext Transport Protocol that uses paired network layer addressing and transport layer ports in request/response, client/server exchanges substantially equivalent to that of Hypertext Transport Protocol.

23. A caching system for a star topology network, according to claim 14, 15, 16, 17, 18, or 19, wherein caching through the second communications path to one or more global caches of specified URLs, or payloads of HTTP messages with specified IP addresses, is blocked or authorized by a content authorization program implemented through the first communications path.

24. A caching system for a star topology network, according to claim 14, 15, 16, 17, 18, or 19, wherein the HTTP responses received through the second communications path are

examined by said buffering component associated with an intermediate node or an end-user node, and if said buffering component detects one or more errors in the payload of an HTTP response, said buffering component discards each HTTP response containing such error(s) and the corresponding buffered HTTP request.

25. A caching system for a star topology network, according to claim 14, 15, 16, 17, 18, or 19, wherein there is a means by which HTTP responses received through the second communications path are examined by said buffering component associated with an intermediate node or an end-user node, and if said buffering component detects an error in the payload of an HTTP response and said buffering component is associated with or interfaced with an inbound channel, the buffering component extracts the relevant URL from the corresponding decapsulated and buffered HTTP request, prepares an HTTP request for such URL, and causes the transmission of the prepared HTTP request in such inbound channel.

26. A caching system for a star topology network, according to claim 14, 15, or 16, further comprising:

a means by which each HTTP request inbound from a remote terminal is relayed to a server computer and a copy of said HTTP request is encapsulated in a UDP datagram at a hub node and distributed through the first communications path to all secondary monitoring components associated with or interfaced with an intermediate node or an end-user node in the star topology network, each of which secondary monitoring component forwards such encapsulated HTTP request to a decapsulating component associated with such secondary monitoring component, such decapsulating component decapsulates such HTTP request and forwards such decapsulated HTTP request to a buffering component associated with such decapsulating component, and such decapsulated HTTP request is tagged by the buffering component based on source socket and destination socket and buffered; and

one or more remote terminals, each of which remote terminals is associated with or interfaced with an intermediate node or end-user node and has a means to output each HTTP response received through the second communications path to a secondary monitoring component associated with or interfaced with such intermediate node or end-user node, wherein such secondary monitoring component forwards such HTTP response to a buffering component associated with such secondary monitoring component, such buffering component examines the source socket and destination socket of such HTTP response and the source socket and destination socket of each HTTP request that has been buffered and either (a) if the buffering component matches such HTTP response with a buffered HTTP request that contains the corresponding source socket and destination socket (the source and destination sockets are reversed in the HTTP request as compared

with the HTTP response), then the buffering component stores in a global cache associated with or interfaced with the buffering component and accessible by one or more end-users the contents of the HTTP response under the URL contained in the HTTP request, or (b) if the buffering component can not match such HTTP response with a buffered HTTP request that contains a corresponding source socket and destination socket, such HTTP response is tagged by the buffering component based on source socket and destination socket of said HTTP response and buffered, then periodically thereafter the buffering component attempts to match the source socket and destination socket of such buffered HTTP response with subsequently buffered HTTP requests and upon successfully matching the sockets of such HTTP response with the sockets of an HTTP request (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response), then the buffering component stores in a global cache associated with or interfaced with the buffering component and accessible by one or more end-users the contents of the HTTP response under the URL extracted from in the HTTP request, and if the sockets of an HTTP response cannot be matched before the expiration of an operator-defined time, the buffering component discards such HTTP response; if the sockets of an HTTP request cannot be matched before the expiration of an operator-defined time, the buffering component discards such HTTP request unless the buffering component has access to a forward communications channel and in which case issues an HTTP request containing the URL in the unmatched and buffered HTTP request.

27. A caching system for a star topology network, according to claim 17, 18, or 19, further comprising:

a means by which each HTTP request inbound from a remote terminal is relayed to a server computer and a copy of such HTTP request is encapsulated in a UDP datagram at said hub node and buffered in a primary buffering component pending the arrival at the hub node of a corresponding HTTP response, and upon arrival of such corresponding HTTP response, said HTTP response is buffered in said primary buffering component and said primary buffering component examines the source socket and destination socket of such HTTP response and the source socket and destination socket of each HTTP request that has been encapsulated and buffered and either (a) if the primary buffering component matches such buffered HTTP response with an encapsulated and buffered HTTP request that contains the corresponding source socket and destination socket (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response), then the primary buffering component forwards the encapsulated and matched HTTP request through the first communications path to all secondary monitoring components associated with or interfaced with an intermediate node or an end-user node in the star topology network, each of which secondary monitoring component forwards

such encapsulated HTTP request to a decapsulating component associated with such secondary monitoring component, such decapsulating component decapsulates such HTTP request and forwards such decapsulated HTTP request to a secondary buffering component associated with such decapsulating component, and such decapsulated HTTP request is tagged by the secondary buffering component based on source socket and destination socket and buffered, or (b) if the primary buffering component can not match an encapsulated buffered HTTP request with a buffered HTTP response that contains a corresponding source socket and destination socket, the primary buffering component either issues another HTTP request, without encapsulation, addressed to the relevant server computer and receives a corresponding HTTP response that the secondary buffering component successfully socket-matches, or discards such encapsulated and buffered HTTP request, as determined by operator-defined configuration settings, and only after forwarding of an encapsulated HTTP request through the first communications path under (a) above, the primary buffering component at the hub node forwards the matching HTTP response through the second communications channel, and

a means associated with or interfaced with each intermediate node or end-user node that outputs each HTTP response received through the second communications path to a secondary monitoring component associated with or interfaced with such intermediate node or end-user node, wherein such secondary monitoring component forwards such HTTP response to a secondary buffering component associated with such secondary monitoring component, such secondary buffering component examines the source socket and destination socket of such HTTP response and the source socket and destination socket of each HTTP request that has been buffered and either (a) if the secondary buffering component matches such HTTP response with a buffered HTTP request that contains the corresponding source socket and destination socket (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response), then the secondary buffering component stores in a global cache associated with or interfaced with the secondary buffering component and accessible by one or more end-users the contents of the HTTP response under the URL contained in the HTTP request, or (b) if the secondary buffering component can not match such HTTP response with a buffered HTTP request that contains a corresponding source socket and destination socket, such HTTP response is tagged by the secondary buffering component based on source socket and destination socket of said HTTP response and buffered, then periodically thereafter the secondary buffering component attempts to match the source socket and destination socket of such buffered HTTP response with subsequently buffered HTTP requests and upon successfully matching the sockets of such HTTP response with the sockets of an HTTP request (the source and destination sockets are reversed in the HTTP request as compared with the

HTTP response), then the secondary buffering component stores in a global cache associated with or interfaced with the secondary buffering component and accessible by one or more end-users the contents of the HTTP response under the URL extracted from in the HTTP request, and if the sockets of an HTTP response cannot be matched before the expiration of an operator-defined time, the secondary buffering component discards such HTTP response; if the sockets of an HTTP request cannot be matched before the expiration of an operator-defined time, the secondary buffering component discards such HTTP request unless the secondary buffering component has access to a forward communications channel and in which case issues an HTTP request containing the URL in the unmatched and buffered HTTP request.

28. A method of encapsulation and passive listening, comprising:

receiving and buffering on a hub node of a network a message from a first network node to said hub node, encapsulating said message within a broadcast or multicast message, and transmitting said broadcast or multicast message to one or more other network nodes unable to receive the original message transmission from said first network node to said hub node,

receiving said broadcast or multicast message, and unencapsulating and buffering said original message from said first node, when said broadcast or multicast message arrives at each of said one or more other network nodes, and

storing the contents of the decapsulated and buffered message in cache memory or storage accessible by an end-user at each of said one or more other network nodes.

29. A method of encapsulation and passive listening, comprising:

receiving and buffering on a hub node of a network a message from a first network node to said hub node, selecting said message for encapsulation based on one or more criteria, such as being an HTTP request or being addressed to a certain network address or socket, encapsulating said message within a broadcast or multicast message, and transmitting said broadcast or multicast message to one or more other network nodes unable to receive the original message transmission from said first network node to said hub node,

when said broadcast or multicast message arrives at each of said one or more other network nodes, receiving said broadcast or multicast message, and unencapsulating and buffering said original message from said first node, at each of said one or more other network nodes, and

storing the contents of the decapsulated and buffered message in cache memory or storage accessible by an end-user at each of said one or more other network nodes.

30. A method of encapsulation and passive listening, according to claim 28 or 29, wherein said message transmitted from said hub node to said one or more other network nodes is

encapsulated in User Datagram Protocol or a transport layer, connectionless protocol substantially equivalent to User Datagram Protocol.

31. A method of socket matching, comprising:

receiving and buffering on a hub node of a network an HTTP request from a first network node to said hub node, forwarding said HTTP request to a server computer and also copying said HTTP request and transmitting said HTTP request to one or more other network nodes unable to receive the original transmission of said HTTP request from said first network node to said hub node, receiving an HTTP response sent by said server computer in response to said HTTP request, and transmitting said HTTP response to said first network node and to said one or more other network nodes,

at each of said one or more other network nodes, receiving and buffering said HTTP request when it arrives, and receiving and buffering said HTTP response when it arrives, in buffer memory associated with or interfaced with each of said one or more other network nodes,

in said buffer memory at each of said one or more other network nodes, matching the source socket and destination socket of a buffered HTTP request with the source socket and destination socket of a buffered HTTP response (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response) sent by the server computer in response to said HTTP request and received before or after receipt of said HTTP request, and

storing under the URL specified in the matching HTTP request the contents of the matched HTTP response in cache memory accessible to an end-user node, which end user node is associated with or interfaced with one of said one or more other network nodes.

32. A method of socket matching, comprising:

receiving and buffering on a hub node of a network an HTTP request from a first network node to said hub node, forwarding said HTTP request to a server computer and also copying said HTTP request and buffering in buffer memory said copied HTTP request until such time as an HTTP response sent by said server computer in response to said HTTP request is received, buffered, and matched as described below at said hub node

receiving and buffering at said hub node an HTTP response sent by said server computer in response to said HTTP request

in said buffer memory said hub node, matching the source socket and destination socket of a buffered HTTP request with the source socket and destination socket of a buffered HTTP response (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response) sent by the server computer in response to said HTTP request and received before or after receipt of said HTTP request, and

transmitting said matched HTTP request from said hub node to one or more other

network nodes unable to receive the original transmission of said HTTP request from said first network node to said hub node, and transmitting said matching HTTP response to said first network node and to said one or more other network nodes,

at each of said one or more other network nodes, receiving and buffering said HTTP request when it arrives, and receiving and buffering said HTTP response when it arrives, in buffer memory associated with or interfaced with each of said one or more other network nodes,

in said buffer memory at each of said one or more other network nodes, matching the source socket and destination socket of a buffered HTTP request with the source socket and destination socket of a buffered HTTP response (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response) sent by the server computer in response to said HTTP request and received before or after receipt of said HTTP request, and

storing under the URL specified in the matching HTTP request the contents of the matched HTTP response in cache memory accessible to an end-user node, which end user node is associated with or interfaced with one of said one or more other network nodes.

33. A method of socket matching, according to claim 31 or 32, wherein Transmission Control Protocol or a transport layer protocol other than Transmission Control Protocol that uses network addressing and routing capabilities of Internet Protocol, or a protocol other than Hypertext Transport Protocol that uses paired network layer addressing and transport layer ports in request/response, client/server exchanges substantially equivalent to that of Hypertext Transport Protocol, is used.

34. A method of socket matching, according to claim 31 or 32, wherein the network is a star topology network selected from the group consisting of very small aperture satellite, multichannel multipoint distribution service, local multipoint distribution service, instructional television fixed service, digital television broadcasting, and digital cablecasting.

35. A method of socket matching, according to claim 31 or 32, wherein each said copy of an HTTP request is encapsulated in a broadcast or multicast message before transmission from said hub node and is decapsulated upon receipt and prior to buffering at each of said one of more other network nodes.

36. A caching method for a star topology network, comprising:

receiving HTTP requests from one or more remote terminals at a hub node of the star topology network, interfacing said hub node with at least one external server computer, and transmitting UDP datagrams, HTTP requests, and HTTP responses in an outbound channel from said hub node of the star topology network, and

receiving said UDP datagrams, HTTP requests, and HTTP responses at one or more intermediate or end-user nodes associated with or interfaced with a remote terminal of said star topology network, socket-matching said HTTP requests and HTTP responses at said one or more intermediate or end-user nodes, and storing in an end-user accessible cache associated with or interfaced with said one or more intermediate or end-user nodes the contents of a given HTTP response under the URL extracted from a correctly matched HTTP request.

37. A caching method for a star topology network, comprising:

receiving HTTP requests from one or more remote terminals at a hub node of the star topology network, interfacing said hub node with at least one external server computer, socket-matching HTTP requests and HTTP responses at said hub node, and transmitting UDP datagrams, HTTP requests, and HTTP responses in an outbound channel from said hub node of the star topology network to said one or more remote terminals, and

receiving said UDP datagrams, HTTP requests, and HTTP responses at one or more intermediate or end-user nodes associated with or interfaced with a remote terminal of said star topology network, socket-matching said HTTP requests and HTTP responses at said one or more intermediate or end-user nodes, and storing in an end-user accessible cache at said one or more intermediate or end-user nodes the contents of a given HTTP response under the URL extracted from a correctly matched HTTP request.

38. A caching method for a star topology network, comprising:

interfacing a hub node of the star topology network with one or more external server computers, and internal to the star topology network associating with the hub node a primary monitoring component, an encapsulating component, and an outbound channel,

associating with at least one intermediate node or end-user node interfaced with a remote terminal in the star topology network a secondary monitoring component, a decapsulating component, and a socket-matching buffering component,

associating with said buffering component and an end-user node at least one global cache, or where said buffering component is associated with an intermediate node, associating with or interfacing with said buffering component and an intermediate node at least one global cache,

providing communications paths for UDP and HTTP messages from the hub node through the outbound channel to each buffering component,

controlling the writing of files or database records in said global cache by means of said associated or interfaced buffering component,

wherein each global cache is accessible by at least one end-user node and contains information or data from said server computers, and

at least one end-user node is able to exchange HTTP requests and HTTP responses

with said hub node using the star network medium or a different telecommunications medium.

39. A caching method for a star topology network, comprising:

interfacing a hub node of the star topology network with one or more external server computers, and internal to the star topology network associating with the hub node a primary monitoring component, an encapsulating component, a socket-matching primary buffering component, and an outbound channel,

associating with at least one intermediate node or end-user node interfaced with a remote terminal in the star topology network a secondary monitoring component, a decapsulating component, and a socket-matching secondary buffering component,

associating with said secondary buffering component and an end-user node at least one global cache, or where said secondary buffering component is associated with an intermediate node, associating with or interfacing with said secondary buffering component and an intermediate node at least one global cache,

providing communications paths for UDP and HTTP messages from the hub node through the outbound channel to each secondary buffering component,

controlling the writing of files or database records in said global cache by means of said associated or interfaced secondary buffering component,

wherein each global cache is accessible by at least one end-user node and contains information or data from said server computers, and

at least one end-user node is able to exchange HTTP requests and HTTP responses with said hub node using the star network medium or a different telecommunications medium.

40. A caching method for a star topology network, according to claim 36, 37, 38, or 39, wherein Transmission Control Protocol or a transport layer protocol other than Transmission Control Protocol that uses the network addressing and routing capabilities of Internet Protocol is used, or a protocol other than Hypertext Transport Protocol that uses paired network layer addressing and transport layer ports in request/response, client/server exchanges substantially equivalent to such use in Hypertext Transport Protocol is used, or a connectionless protocol other than User Datagram Protocol with broadcast or multicast addressing substantially equivalent to that of User Datagram Protocol, is used.

41. A caching method for a star topology network, comprising:

interfacing a hub node of the star topology network with one or more external server computers, and internal to the star topology network associating with the hub node a primary monitoring component, an encapsulating component, and an outbound channel,

associating with at least one end-user node interfaced with a remote terminal in the star topology network a secondary monitoring component, a decapsulating component, a

socket-matching buffering component, and at least one global cache,

providing a first communications path from the hub node through the primary monitoring component, encapsulating component, and outbound channel, thence through the remote terminal, the secondary monitoring component and decapsulating component to each said buffering component,

providing a second communications path from said one or more external server computers through the outbound channel of said hub node, thence through the remote terminal, the secondary monitoring component to each said buffering component,

controlling the writing of files or database records in said associated global cache by means of said buffering component,

wherein said global cache is accessible by said end-user node and contains information or data from said server computers, and

at least one end-user node is able to exchange messages with said hub node using the star network medium or a different telecommunications medium.

42. A caching method for a star topology network, comprising:

interfacing a hub node of the star topology network with one or more external server computers, and internal to the star topology network associating with the hub node a primary monitoring component, an encapsulating component, and an outbound channel,

associating with at least one intermediate node interfaced with a remote terminal in the star topology network a secondary monitoring component, a decapsulating component, and a socket-matching buffering component,

associating at least one global cache with at least one end-user node interfaced with said buffering component,

providing a first communications path from the hub node through the primary monitoring component, encapsulating component, and outbound channel, thence through the remote terminal, the secondary monitoring component and decapsulating component to each said buffering component,

providing a second communications path from said one or more external server computers through the outbound channel of said hub node, thence through the remote terminal, the secondary monitoring component to each said buffering component,

controlling the writing of files or database records in said interfaced global cache by means of said buffering component,

wherein said global cache is accessible by said end-user node and contains information or data from said server computers, and

at least one end-user node is able to exchange messages with said hub node using the star network medium or a different telecommunications medium.

43. A caching method for a star topology network, comprising:

interfacing a hub node of the star topology network with one or more external server computers, and internal to the star topology network associating with the hub node a primary monitoring component, an encapsulating component, and an outbound channel,

associating with at least one intermediate node interfaced with a remote terminal in the star topology network a secondary monitoring component, a decapsulating component, a socket-matching buffering component, and at least one global cache,

providing a first communications path from the hub node through the primary monitoring component, encapsulating component, and outbound channel, thence through the remote terminal, the secondary monitoring component and decapsulating component to each said buffering component,

providing a second communications path from said one or more external server computers through the outbound channel of said hub node, thence through the remote terminal, the secondary monitoring component to each said buffering component,

controlling the writing of files or database records in said associated global cache by means of said buffering component,

wherein each global cache is accessible by at least one end-user node interfaced with said intermediate node and contains information or data from said server computers, and

at least one end-user node is able to exchange messages with said hub node using the star network medium or a different telecommunications medium.

44. A caching method for a star topology network, comprising:

interfacing a hub node of the star topology network with one or more external server computers, and internal to the star topology network associating with the hub node a primary monitoring component, an encapsulating component, a socket-matching primary buffering component, and an outbound channel,

associating with at least one end-user node interfaced with a remote terminal in the star topology network a secondary monitoring component, a decapsulating component, a socket-matching secondary buffering component, and at least one global cache,

providing a first communications path from the hub node through said primary monitoring component, encapsulating component, primary buffering component, and outbound channel, thence through the remote terminal, said secondary monitoring component and decapsulating component to each said secondary buffering component,

providing a second communications path from said one or more external server computers through the primary monitoring component, primary buffering component, and outbound channel of said hub node, thence through the remote terminal, said secondary monitoring component to each said secondary buffering component,

controlling the writing of files or database records in said associated global cache

by means of said secondary buffering component,

wherein said global cache is accessible by said end-user node and contains information or data from said server computers, and

at least one end-user node is able to exchange messages with said hub node using the star network medium or a different telecommunications medium.

45. A caching method for a star topology network, comprising:

interfacing a hub node of the star topology network with one or more external server computers, and internal to the star topology network associating with the hub node a primary monitoring component, an encapsulating component, a socket-matching primary buffering component, and an outbound channel,

associating with at least one intermediate node interfaced with a remote terminal in the star topology network a secondary monitoring component, a decapsulating component, and a socket-matching secondary buffering component,

associating at least one global cache with at least one end-user node interfaced with said secondary buffering component,

providing a first communications path from the hub node through said primary monitoring component, encapsulating component, primary buffering component, and outbound channel, thence through the remote terminal, said secondary monitoring component and decapsulating component to each said secondary buffering component,

providing a second communications path from said one or more external server computers through the primary monitoring component, primary buffering component, and outbound channel of said hub node, thence through the remote terminal, said secondary monitoring component to each said secondary buffering component,

controlling the writing of files or database records in said interfaced global cache by means of said secondary buffering component,

wherein said global cache is accessible by said end-user node and contains information or data from said server computers, and

at least one end-user node is able to exchange messages with said hub node using the star network medium or a different telecommunications medium.

46. A caching method for a star topology network, comprising:

interfacing a hub node of the star topology network with one or more external server computers, and internal to the star topology network associating with the hub node a primary monitoring component, an encapsulating component, a socket-matching primary buffering component, and an outbound channel,

associating with at least one intermediate node interfaced with a remote terminal in the star topology network a secondary monitoring component, a decapsulating component, a socket-matching secondary buffering component, and at least one global

cache,

providing a first communications path from the hub node through said primary monitoring component, encapsulating component, primary buffering component, and outbound channel, thence through the remote terminal, said secondary monitoring component and decapsulating component to each said secondary buffering component,

providing a second communications path from said one or more external server computers through the primary monitoring component, primary buffering component, and outbound channel of said hub node, thence through the remote terminal, said secondary monitoring component to each said secondary buffering component,

controlling the writing of files or database records in said associated global cache by means of said secondary buffering component,

wherein each global cache is accessible by at least one end-user node interfaced with said intermediate node and contains information or data from said server computers, and

at least one end-user node is able to exchange messages with said hub node using the star network medium or a different telecommunications medium.

47. A caching method for a star topology network, according to claim 36, 37, 38, 40, 41, 42, 43, 44, 45, or 46, wherein the star networking technology is very small aperture satellite ("VSAT"), multichannel multipoint distribution service ("MMDS"), local multipoint distribution service ("LMDS"), instructional television fixed service ("ITFS"), digital television broadcasting, or digital cablecasting.

48. A caching method for a star topology network, according to claim 36, 37, 38, 40, 41, 42, 43, 44, 45, or 46, wherein a Passive Internet Cache system concurrently uses one or more wireline outbound channels, wireless outbound channels, wireline inbound channels, and/or wireless inbound channels.

49. A caching method for a star topology network, according to claim 41, 42, 43, 44, 45, or 46, wherein the network layer protocol is Internet Protocol or a protocol that uses network addressing and routing substantially equivalent to that of Internet Protocol is used, the transport layer protocol used in the first communications path to encapsulate an HTTP request is User Datagram Protocol or a connectionless protocol substantially equivalent to that of User Datagram Protocol is used, the transport layer protocol used in the second communications path is Transport Control Protocol or a connection-oriented protocol substantially equivalent to Transport Control Protocol is used, or the applications and services protocol is a protocol other than Hypertext Transport Protocol that uses paired network layer addressing and transport layer ports in request/response, client/server exchanges substantially equivalent to that of Hypertext Transport Protocol.

50. A caching method for a star topology network, according to claim 41, 42, 43, 44, 45,

or 46, wherein caching through the second communications path to one or more global caches of specified URLs, or payloads of HTTP messages with specified IP addresses, is blocked or authorized by a content authorization program implemented through the first communications path.

51. A caching method for a star topology network, according to claim 41, 42, 43, 44, 45, or 46, wherein the HTTP responses received through the second communications path are examined by said buffering component associated with an intermediate node or an end-user node, and if said buffering component detects one or more errors in the payload of an HTTP response, said buffering component discards each HTTP response containing such error(s) and the corresponding buffered HTTP request.

52. A caching method for a star topology network, according to claim 41, 42, 43, 44, 45, or 46, wherein there is a means by which HTTP responses received through the second communications path are examined by said buffering component associated with an intermediate node or an end-user node, and if said buffering component detects an error in the payload of an HTTP response and said buffering component is associated with or interfaced with an inbound channel, the buffering component extracts the relevant URL from the corresponding decapsulated and buffered HTTP request, prepares an HTTP request for such URL, and causes the transmission of the prepared HTTP request in such inbound channel.

53. A caching method for a star topology network, according to claim 41, 42, or 43, further comprising:

relaying each HTTP request inbound from a remote terminal through the hub node to a server computer and encapsulating a copy of each said HTTP request in a UDP datagram at said hub node, distributing said UDP datagram through the first communications path to all secondary monitoring components associated with or interfaced with an intermediate node or an end-user node in the star topology network, forwarding such encapsulated HTTP request from each such secondary monitoring component to a decapsulating component associated with each such secondary monitoring component, decapsulating such HTTP request in the decapsulating component, forwarding such decapsulated HTTP request from such decapsulating component to a buffering component associated with such decapsulating component, and tagging by such buffering component of each such decapsulated HTTP request based on source socket and destination socket and buffered; and

associating with or interfacing with one or more remote terminals an intermediate node or end-user node, outputting from each such remote terminal each HTTP response received through the second communications path to a secondary monitoring component associated with or interfaced with such intermediate node or end-user node, forwarding

from the secondary monitoring component such HTTP response to a buffering component associated with such secondary monitoring component, examining the source socket and destination socket of such HTTP response and the source socket and destination socket of each HTTP request that has been buffered by such buffering component, and either (a) if the buffering component matches such HTTP response with a buffered HTTP request that contains the corresponding source socket and destination socket (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response), then storing by the buffering component in a global cache associated with or interfaced with the secondary buffering component and accessible by one or more end-users the contents of the HTTP response under the URL contained in the HTTP request, or (b) if the buffering component can not match such HTTP response with a buffered HTTP request that contains a corresponding source socket and destination socket, tagging such HTTP response by the buffering component based on source socket and destination socket of said HTTP response and buffering such tagged HTTP response, then periodically thereafter attempting by the buffering component to match the source socket and destination socket of such buffered HTTP response with subsequently buffered HTTP requests and upon successfully matching the sockets of such HTTP response with the sockets of an HTTP request (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response), storing by the buffering component in a global cache associated with or interfaced with the secondary buffering component and accessible by one or more end-users the contents of the HTTP response under the URL extracted from in the HTTP request, and if the sockets of an HTTP response cannot be matched before the expiration of an operator-defined time, discarding by the buffering component of such HTTP response; if the sockets of an HTTP request cannot be matched before the expiration of an operator-defined time, discarding by the buffering component of such HTTP request unless the buffering component has access to a forward communications channel and in which case issuing by the buffering component of an HTTP request containing the URL in the unmatched and buffered HTTP request.

54. A caching method for a star topology network, according to claim 44, 45, or 46, further comprising:

relaying each HTTP request inbound from a remote terminal through the hub node to a server computer and encapsulating a copy of each said HTTP request in a UDP datagram at said hub node, buffering said UDP datagram in a primary buffering component pending the arrival at the hub node of a corresponding HTTP response, and upon arrival of such corresponding HTTP response, buffering said HTTP response in said primary buffering component, examining by said primary buffering component of the source socket and destination socket of such HTTP response and the source socket and destination socket of

each HTTP request that has been encapsulated and buffered and either (a) if the primary buffering component matches such buffered HTTP response with an encapsulated and buffered HTTP request that contains the corresponding source socket and destination socket (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response), forwarding by the primary buffering component of the encapsulated and matched HTTP request through the first communications path to all secondary monitoring components associated with or interfaced with an intermediate node or an end-user node in the star topology network, forwarding by each such secondary monitoring components of such encapsulated HTTP request to a decapsulating component associated with such secondary monitoring component, decapsulating by the decapsulating component of such HTTP request, forwarding such decapsulated HTTP request to a secondary buffering component associated with such decapsulating component, and tagging of such decapsulated HTTP request by the secondary buffering component based on source socket and destination socket and buffered, or (b) if the primary buffering component cannot match an encapsulated buffered HTTP request with a buffered HTTP response that contains a corresponding source socket and destination socket, either issuing by the primary buffering component of another HTTP request, without encapsulation, addressed to the relevant server computer and receiving a corresponding HTTP response that the secondary buffering component successfully socket-matches, or discarding such encapsulated and buffered HTTP request, as determined by operator-defined configuration settings, and only after forwarding of an encapsulated HTTP request through the first communications path under (a) above, forwarding by the primary buffering component at the hub node of the matching HTTP response through the second communications channel, and

associating with or interfacing with one or more remote terminals an intermediate node or end-user node, outputting from each such remote terminal each HTTP response received through the second communications path to a secondary monitoring component associated with or interfaced with such intermediate node or end-user node, forwarding from the secondary monitoring component such HTTP response to a secondary buffering component associated with such secondary monitoring component, examining the source socket and destination socket of such HTTP response and the source socket and destination socket of each HTTP request that has been buffered by such secondary buffering component, and either (a) if the secondary buffering component matches such HTTP response with a buffered HTTP request that contains the corresponding source socket and destination socket (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response), then storing by the secondary buffering component in a global cache associated with or interfaced with the secondary buffering component and

accessible by one or more end-users the contents of the HTTP response under the URL contained in the HTTP request, or (b) if the secondary buffering component can not match such HTTP response with a buffered HTTP request that contains a corresponding source socket and destination socket, tagging such HTTP response by the secondary buffering component based on source socket and destination socket of said HTTP response and buffering such tagged HTTP response, then periodically thereafter attempting by the secondary buffering component to match the source socket and destination socket of such buffered HTTP response with subsequently buffered HTTP requests and upon successfully matching the sockets of such HTTP response with the sockets of an HTTP request (the source and destination sockets are reversed in the HTTP request as compared with the HTTP response), storing by the secondary buffering component in a global cache associated with or interfaced with the secondary buffering component and accessible by one or more end-users the contents of the HTTP response under the URL extracted from in the HTTP request, and if the sockets of an HTTP response cannot be matched before the expiration of an operator-defined time, discarding by the secondary buffering component of such HTTP response; if the sockets of an HTTP request cannot be matched before the expiration of an operator-defined time, discarding by the secondary buffering component of such HTTP request unless the secondary buffering component has access to a forward communications channel and in which case issuing by the secondary buffering component of an HTTP request containing the URL in the unmatched and buffered HTTP request.

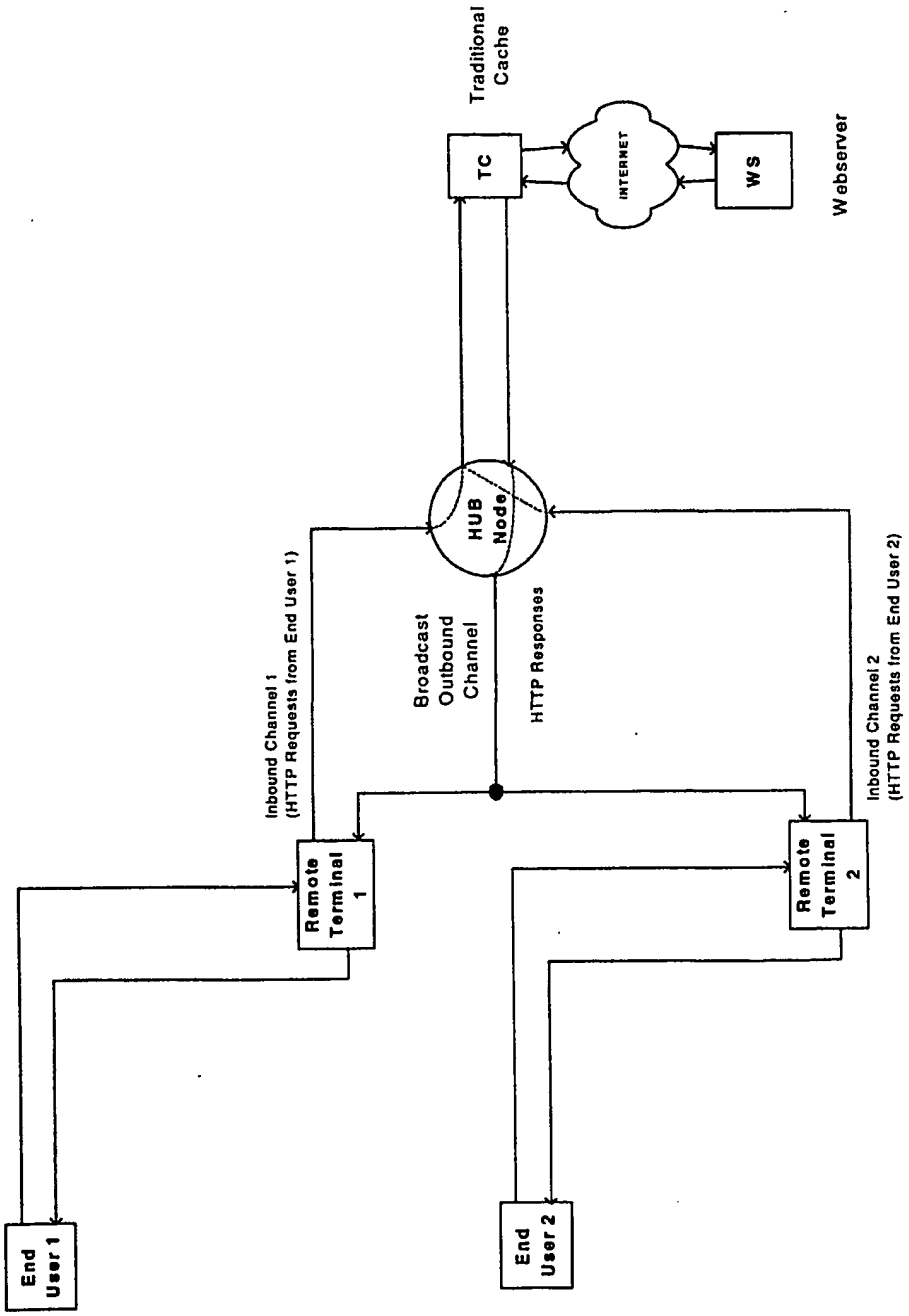


Figure 1. Current Caching In a Star Topology, Dual Channel Network

Figure 1A - End-User Exchange (URL in Cache)

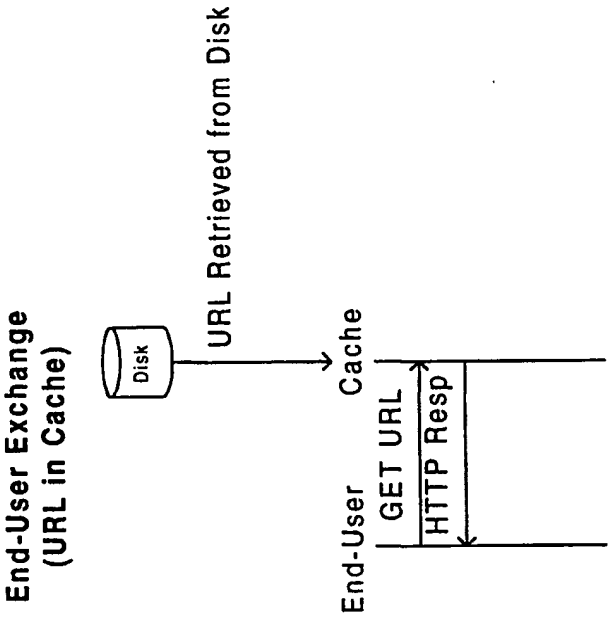


Figure 1B - End-User Exchange (URL not in LOcal or Hub Cache)

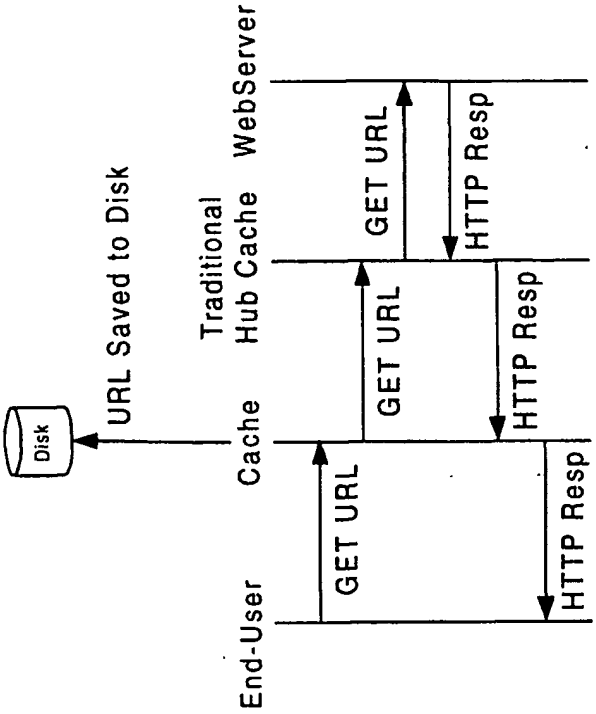
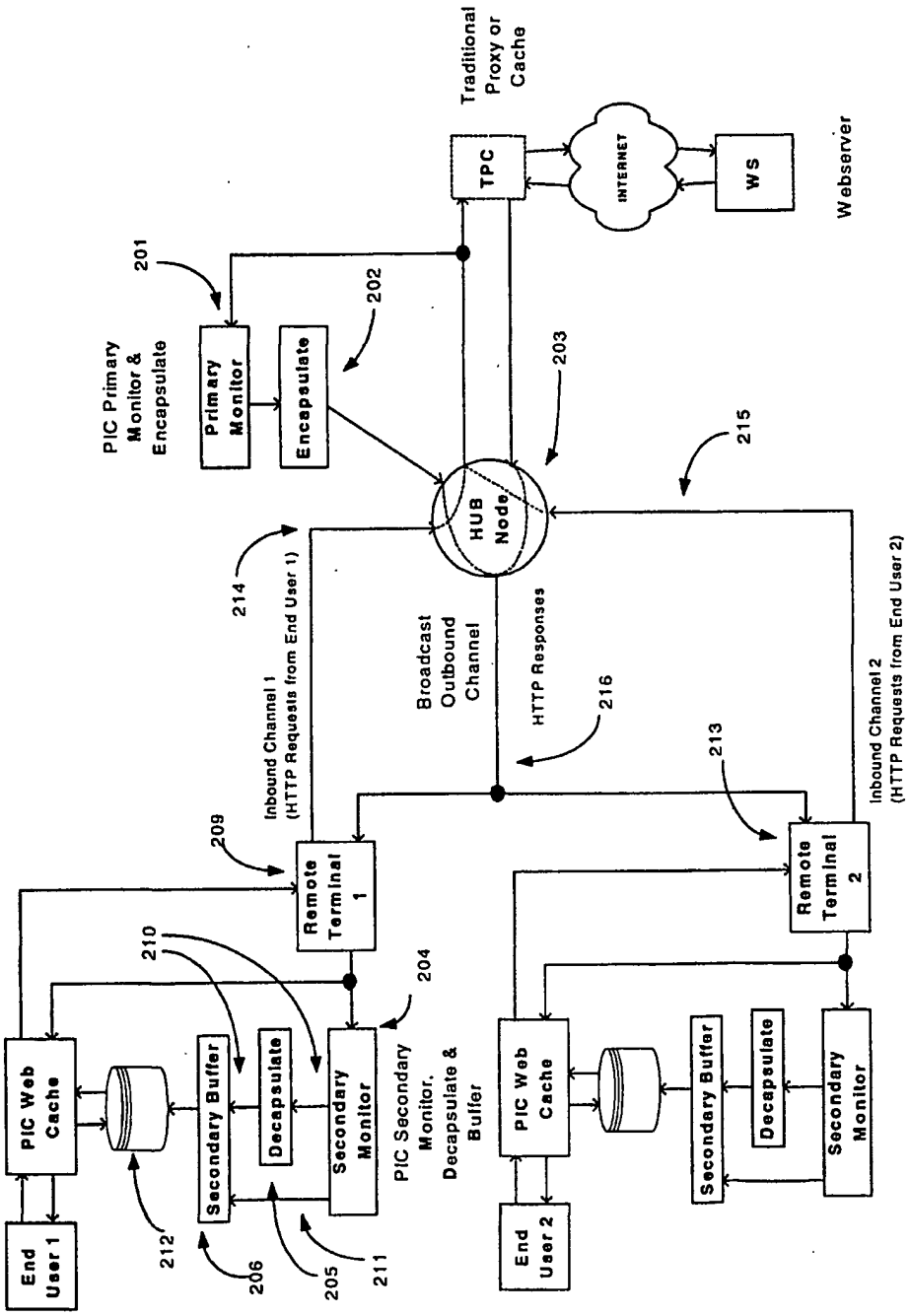


Figure 2. Passive Internet Cache with Socket Matching and PIC Webcache at an End-User Node (First Embodiment)



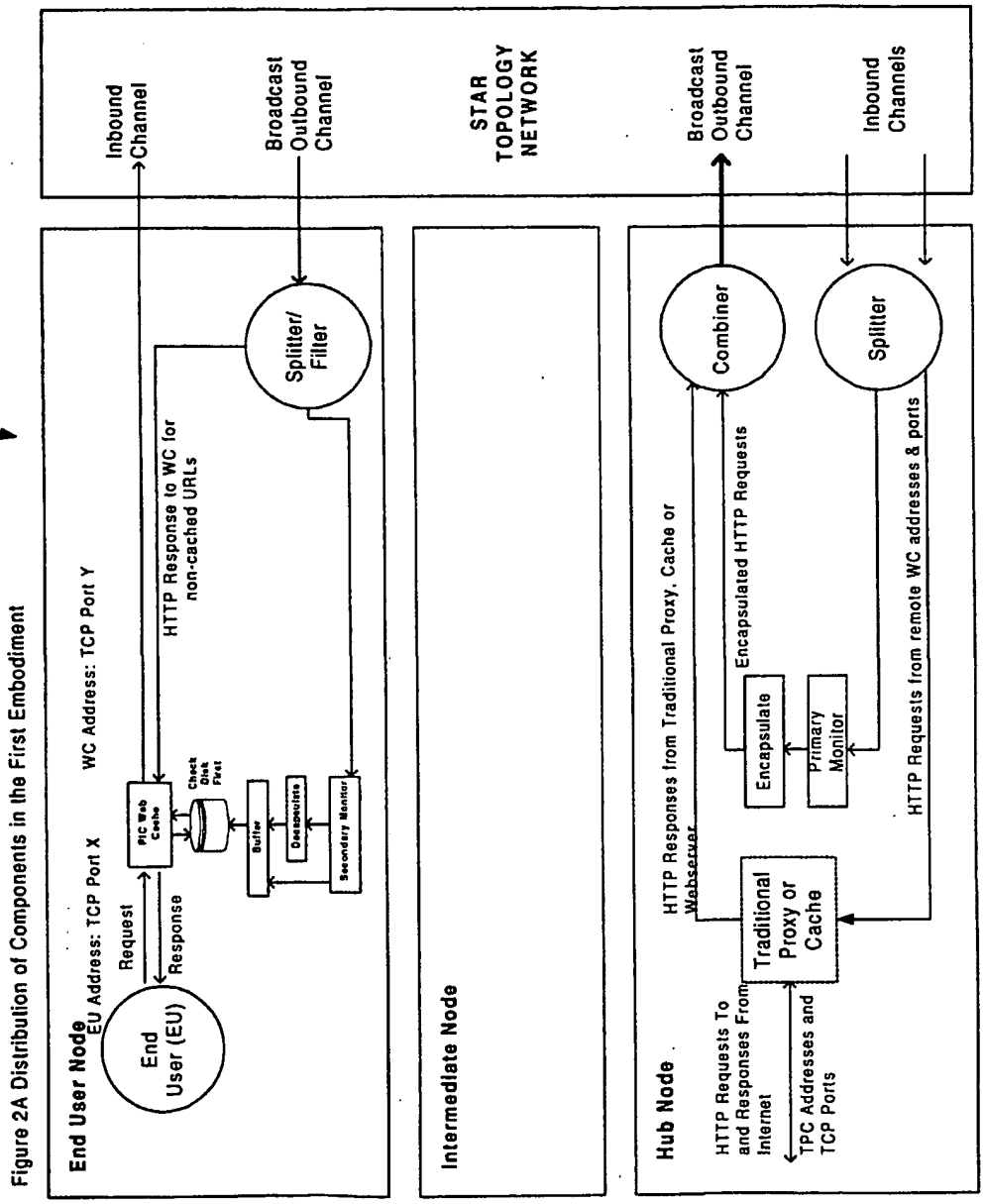


Figure 2A Distribution of Components in the First Embodiment

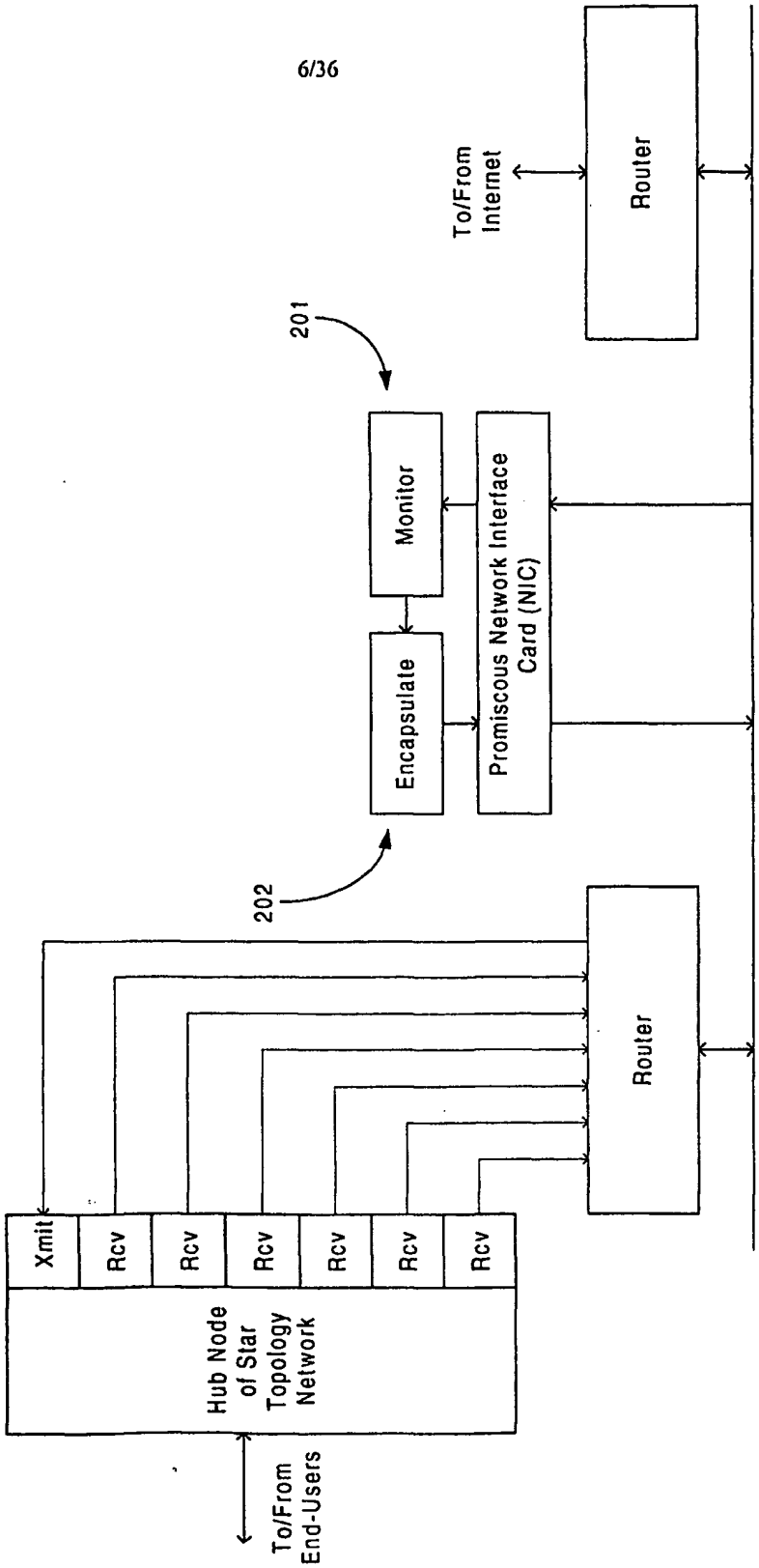


Figure 2B - Passive Listening and Encapsulation at Hub Node

Figure 2C - Socket Matching

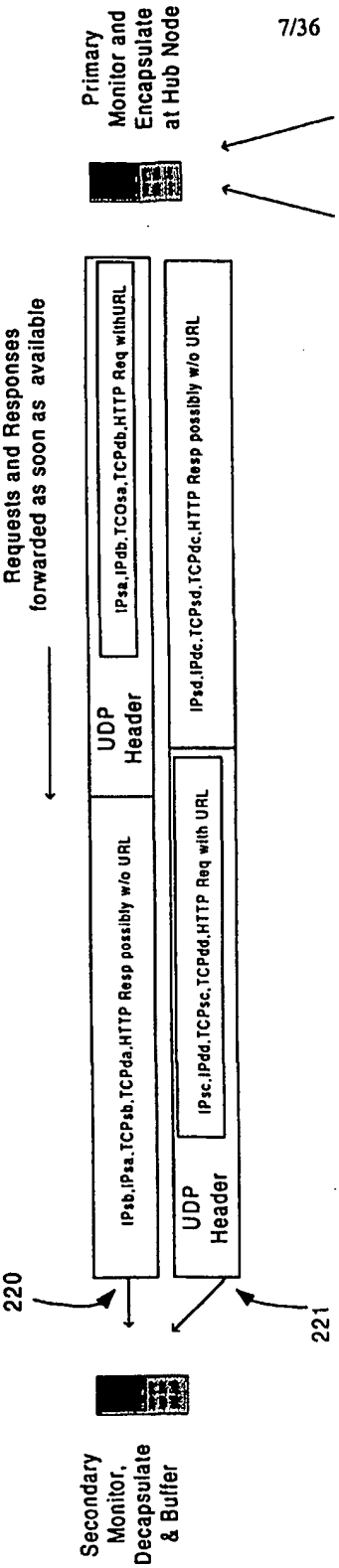
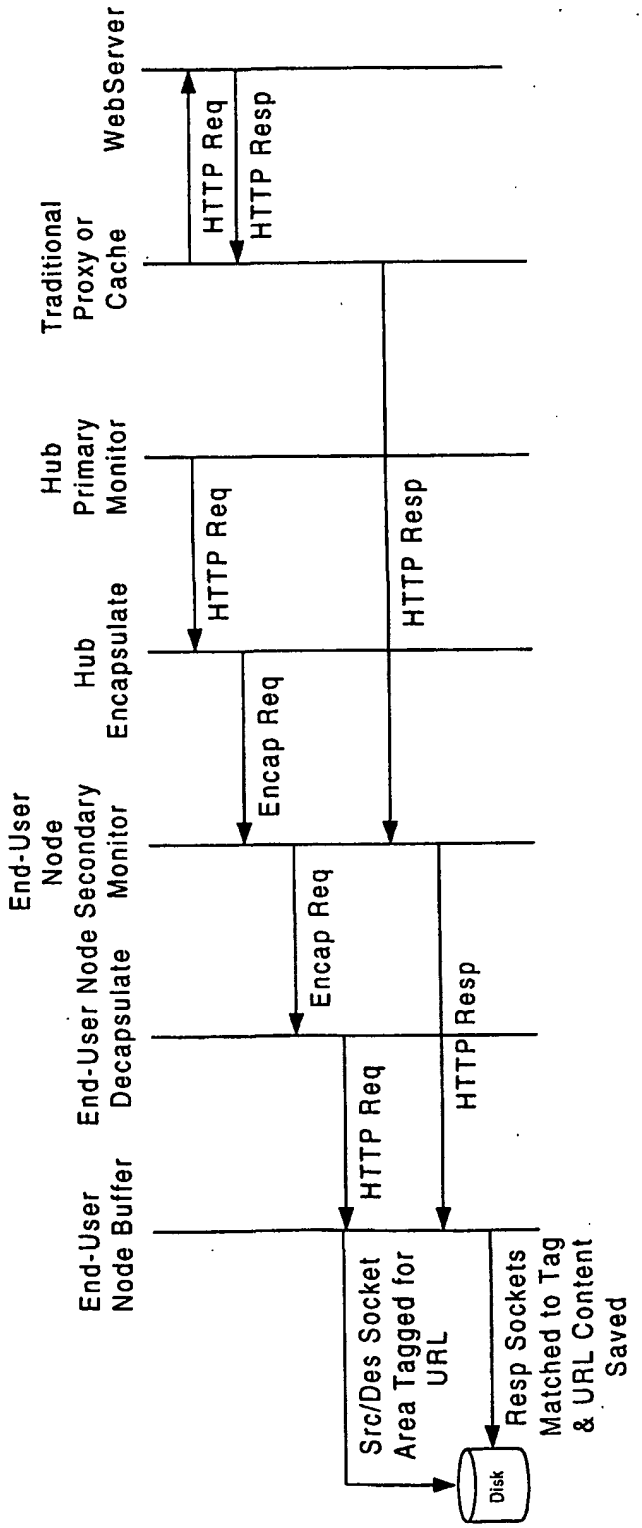


Figure 2D - Embodiment 1 Message Exchange



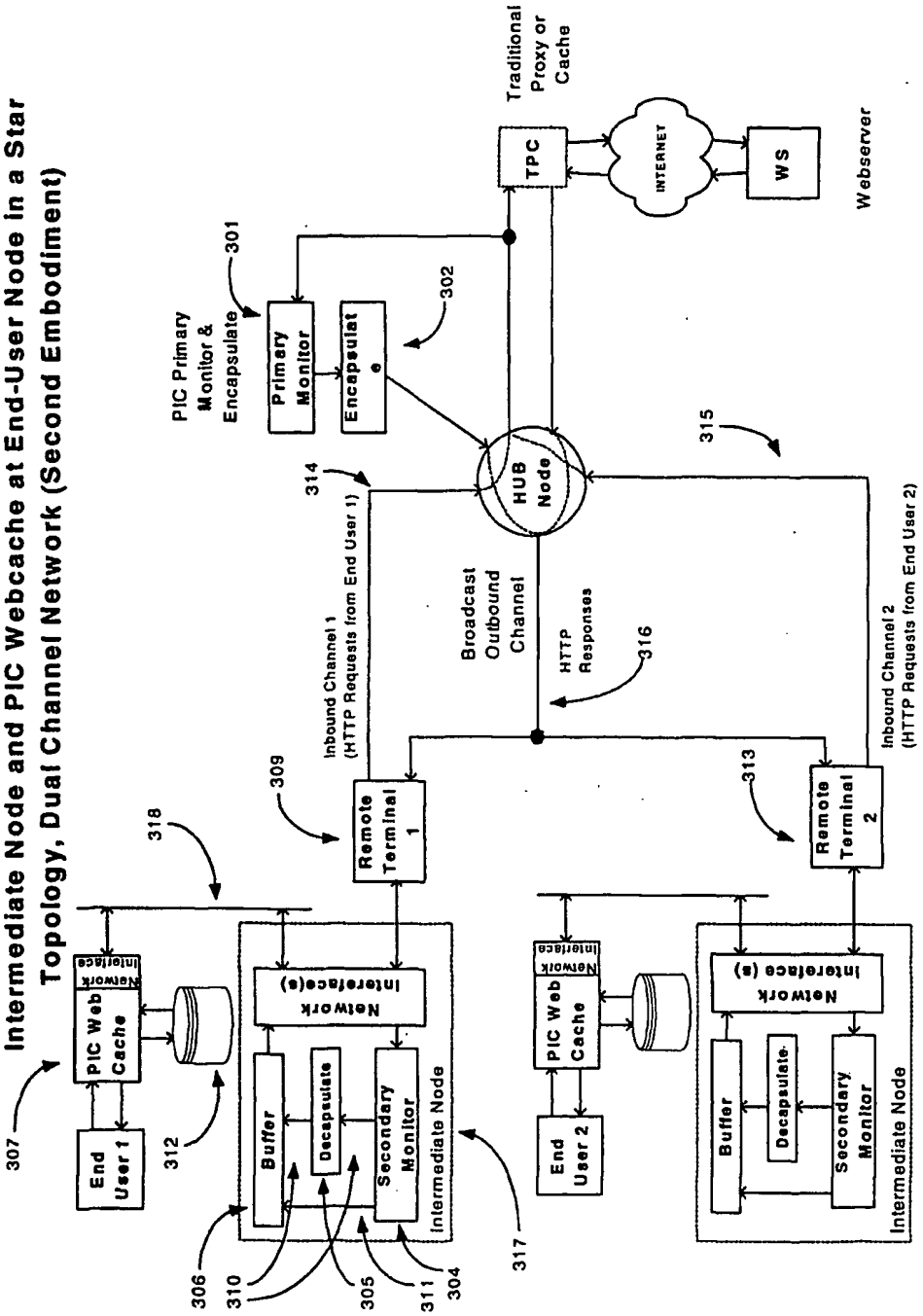


Figure 3. Passive Internet Cache with Socket Matching at Intermediate Node and PIC Webcache at End-User Node in a Star Topology, Dual Channel Network (Second Embodiment)

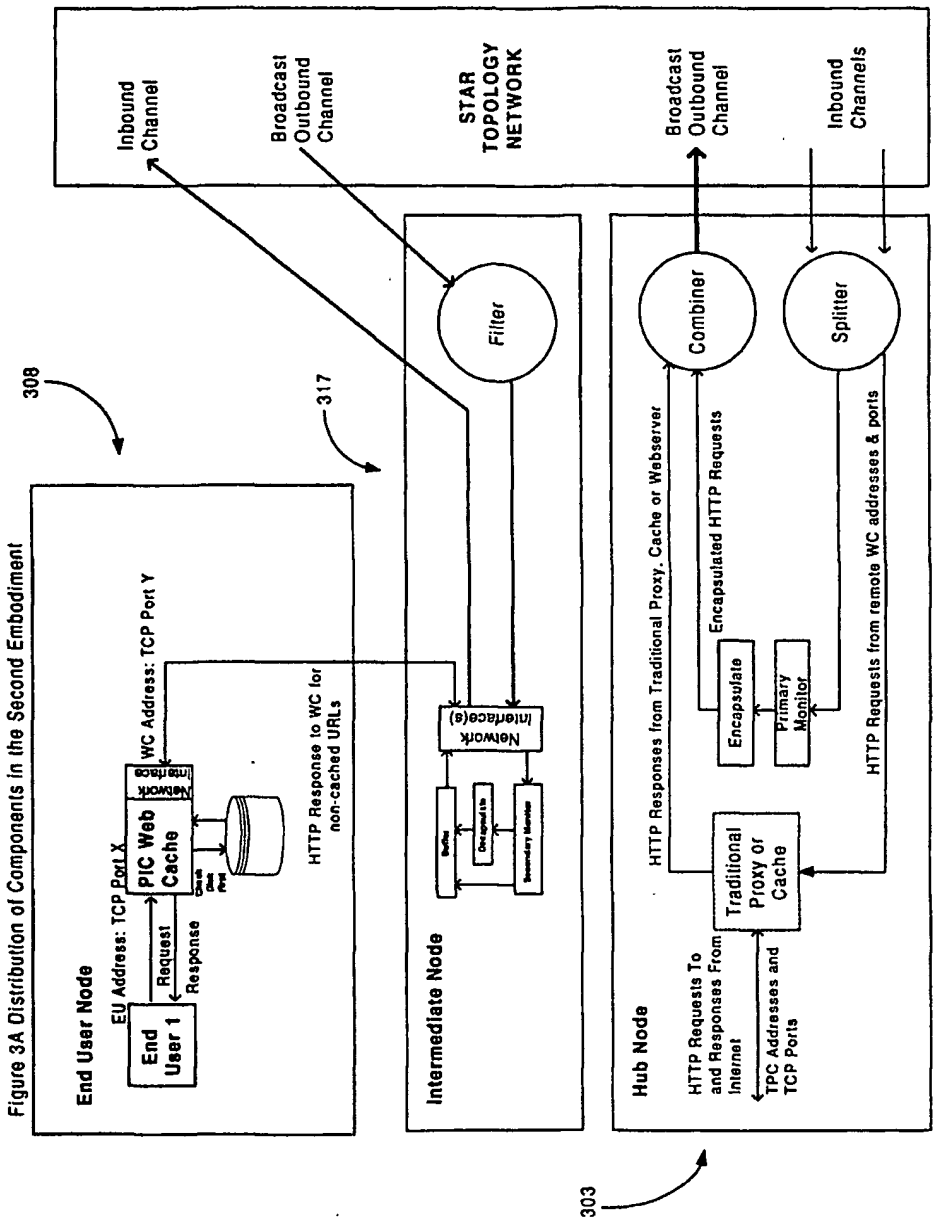
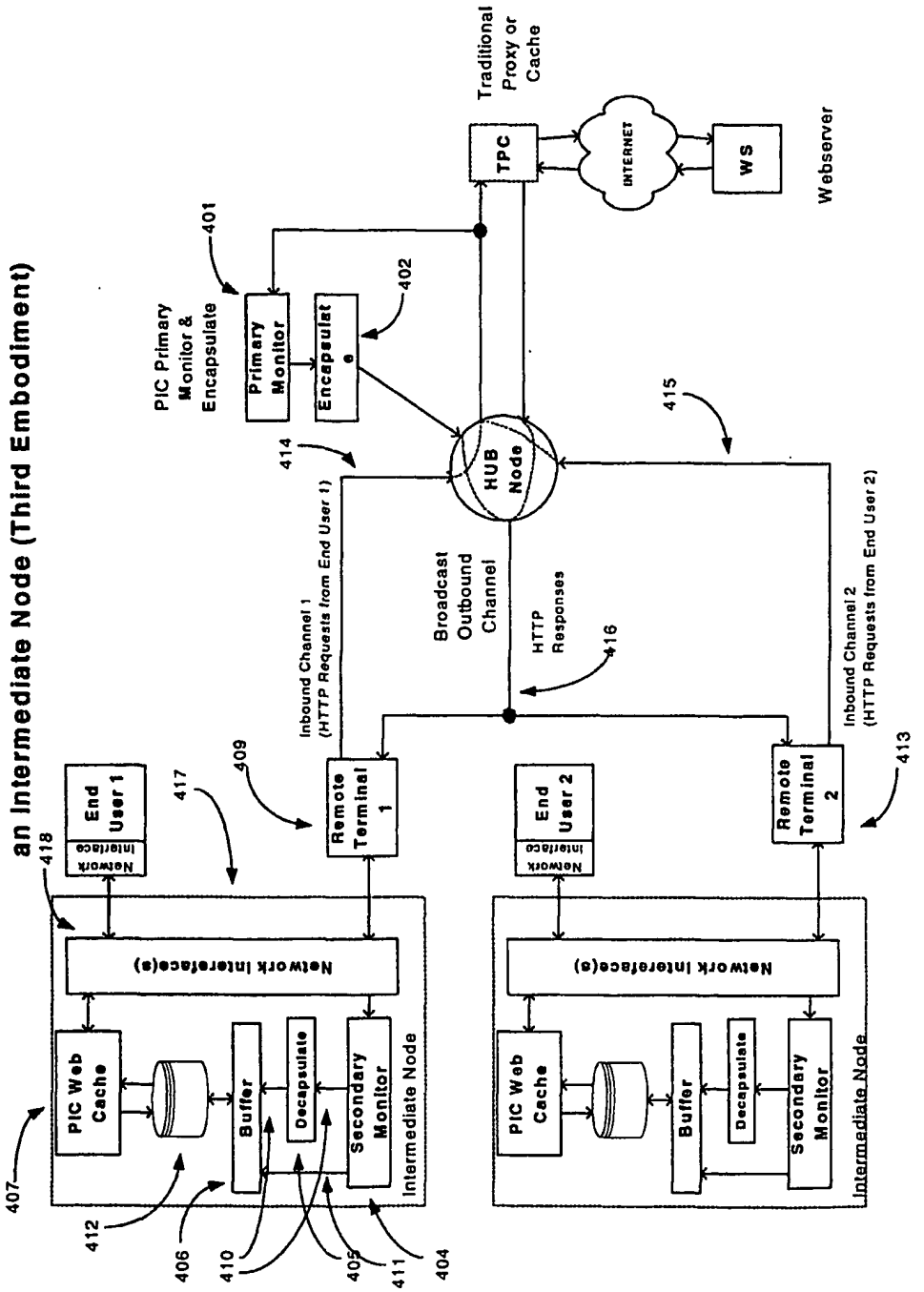


Figure 3A Distribution of Components in the Second Embodiment

Figure 4. Passive Internet Cache with Socket Matching and PIC Webcache at an Intermediate Node (Third Embodiment)



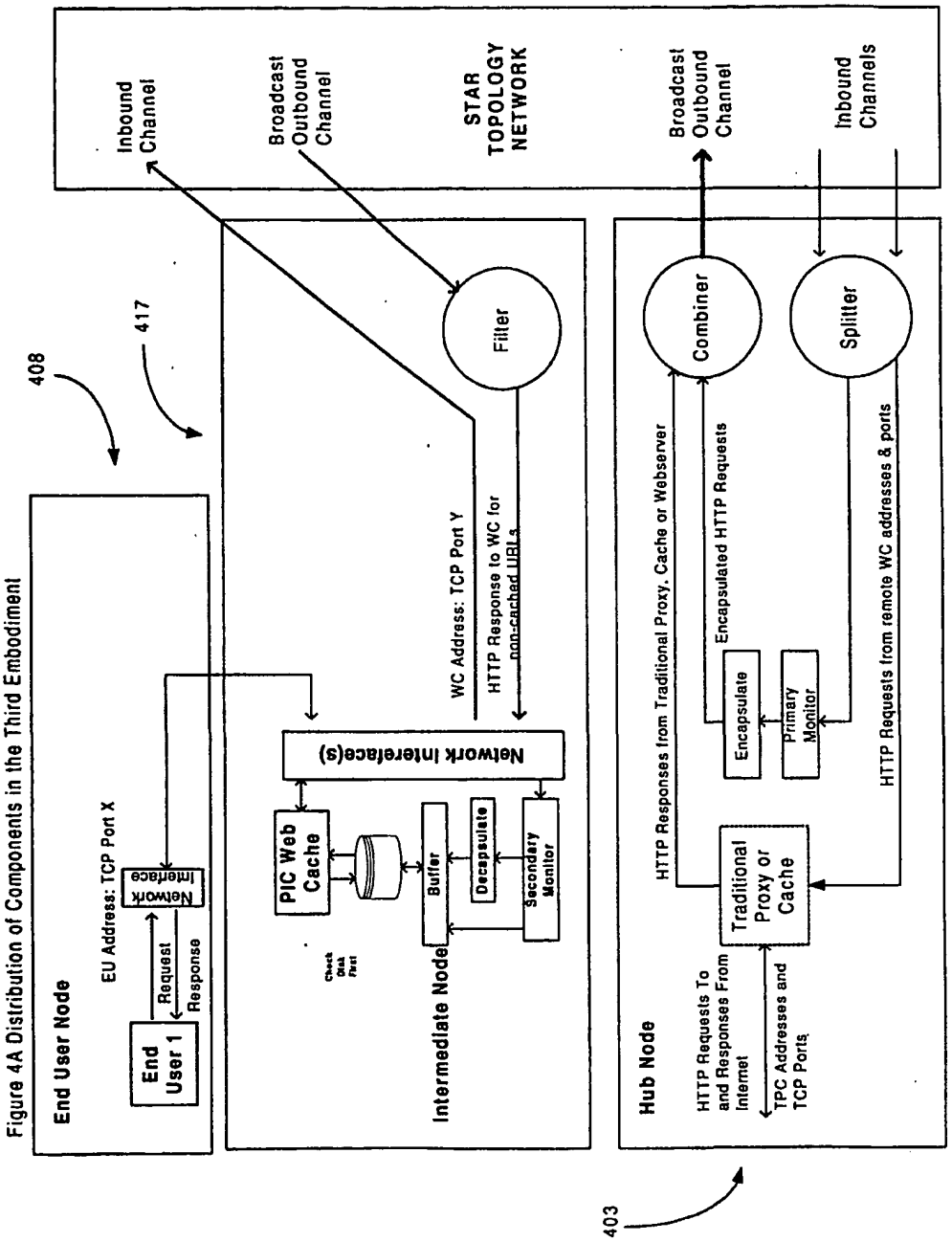


Figure 4A Distribution of Components in the Third Embodiment

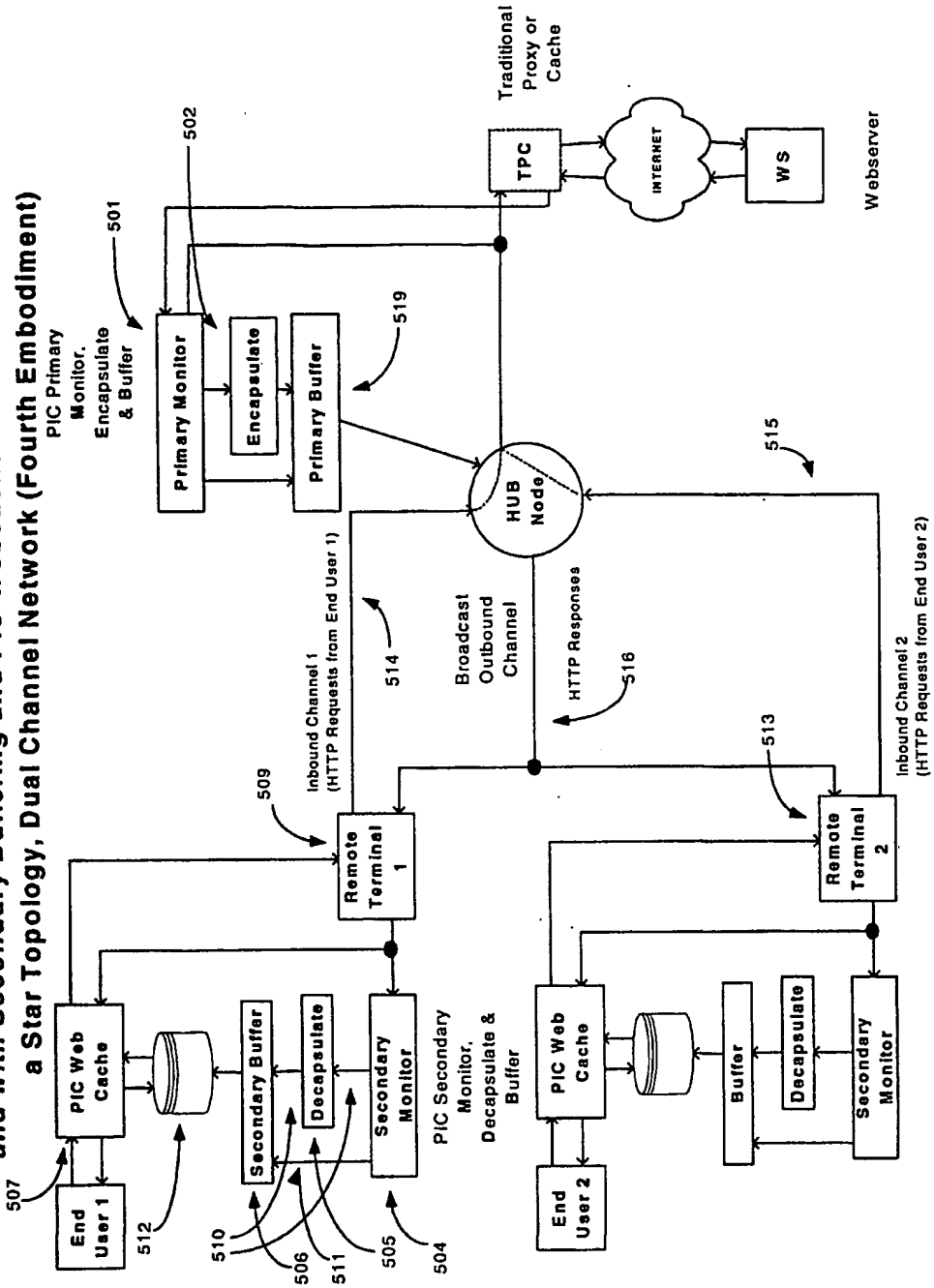
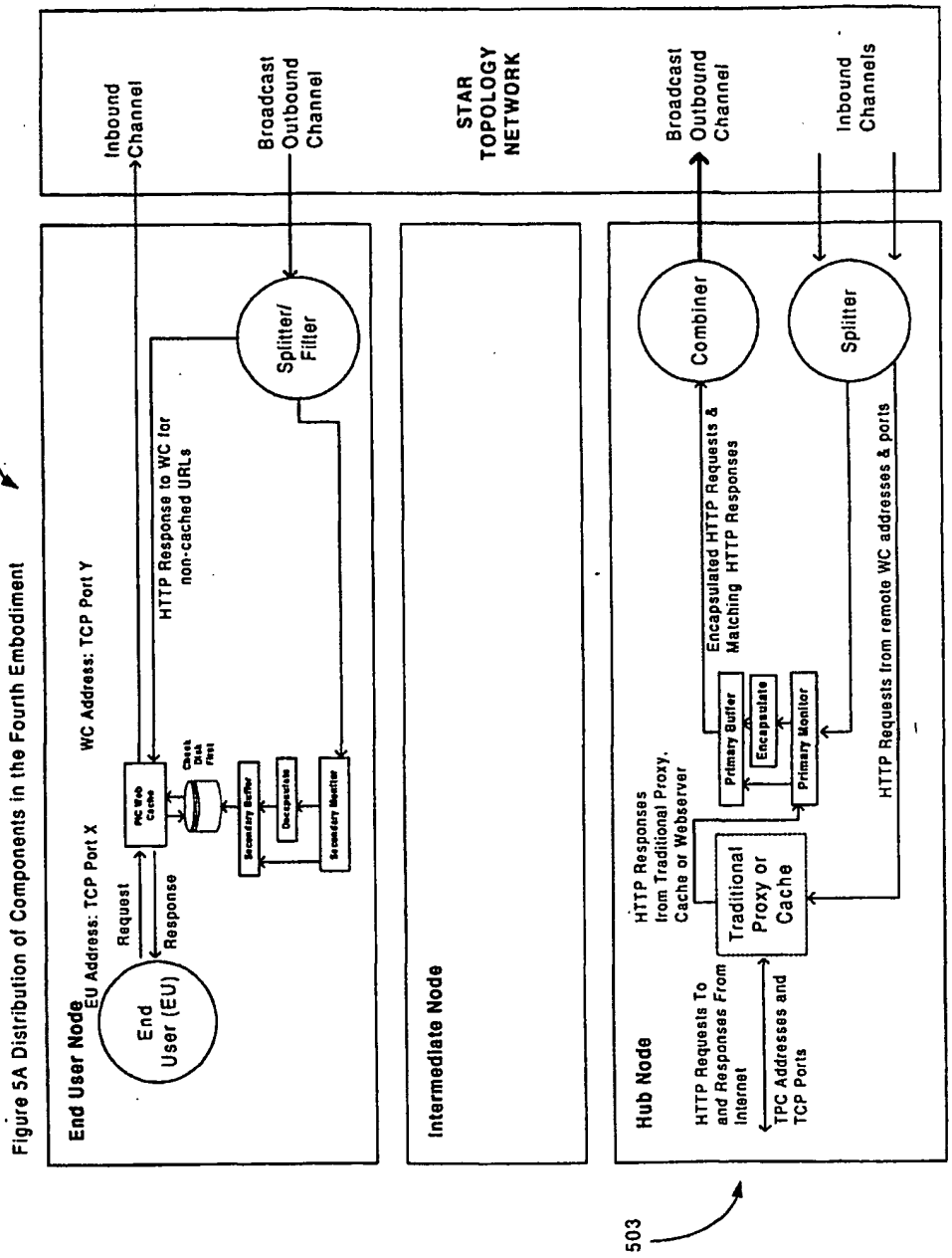


Figure 5. Passive Internet Cache with Socket Matching at the Hub Node and with Secondary Buffering and PIC Webcache at an End-User Node in a Star Topology, Dual Channel Network (Fourth Embodiment)


508
Figure 5A Distribution of Components in the Fourth Embodiment
End User Node
EU Address: TCP Port X
WC Address: TCP Port Y
End User (EU)
Request
Response
PC Web Cache
Check Disk First
Secondary Buffer
Decapsulate
Secondary Monitor
HTTP Response to WC for non-cached URLs
Splitter/ Filter
Inbound Channel
Broadcast Outbound Channel
STAR TOPOLOGY NETWORK
Intermediate Node
Hub Node
HTTP Responses from Traditional Proxy, Cache or Webserver
Traditional Proxy or Cache
HTTP Requests To and Responses From Internet
TPC Addresses and TCP Ports
Primary Buffer
Encapsulate
Primary Monitor
Encapsulated HTTP Requests & Matching HTTP Responses
Combiner
Splitter
HTTP Requests from remote WC addresses & ports
Broadcast Outbound Channel
Inbound Channels
503

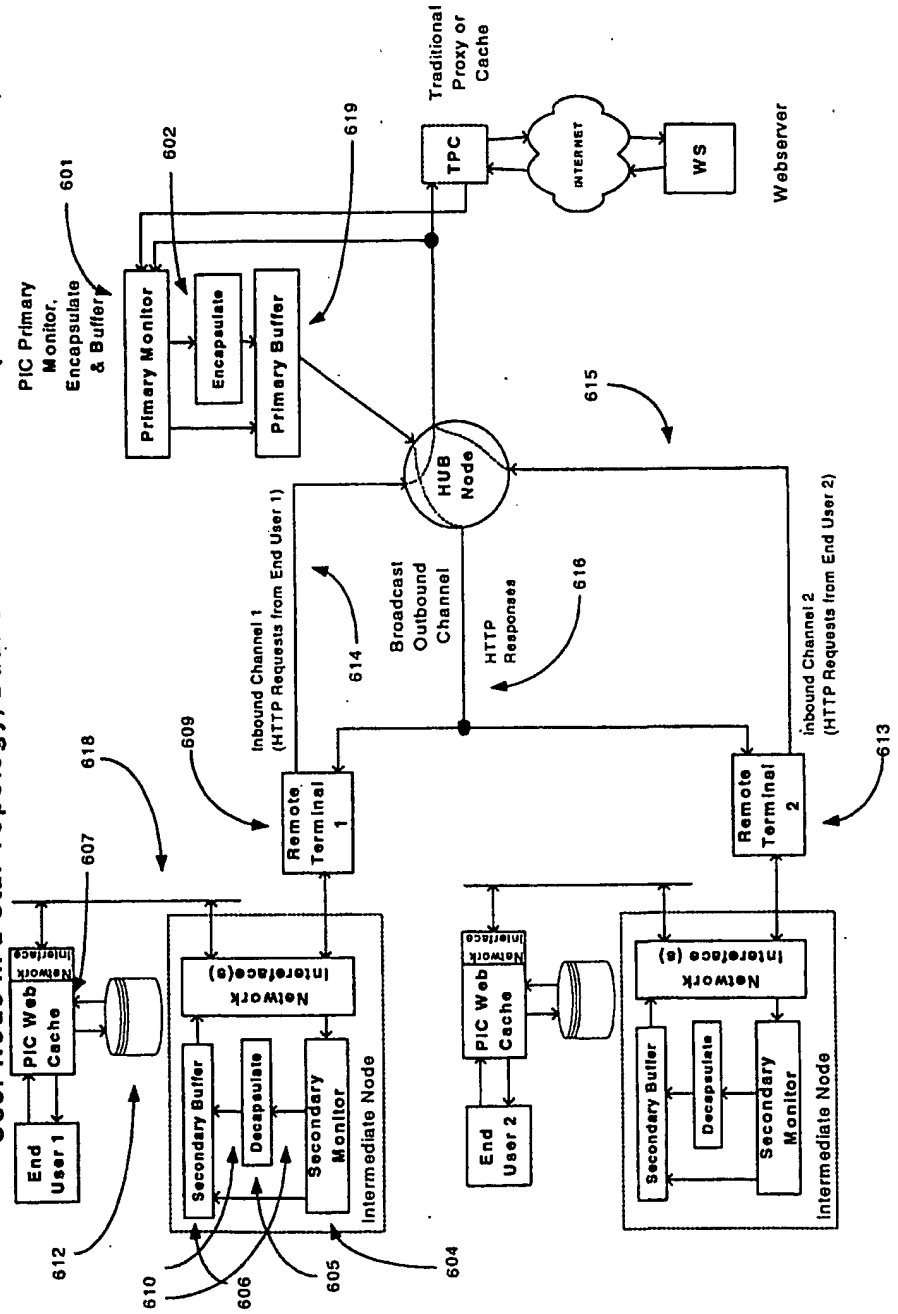


Figure 6. Passive Internet Cache with Socket Matching at the Hub Node, with Secondary Buffering at an Intermediate Node, and with PIC Webcache at End-User Node in a Star Topology, Dual Channel Network (Fifth Embodiment)

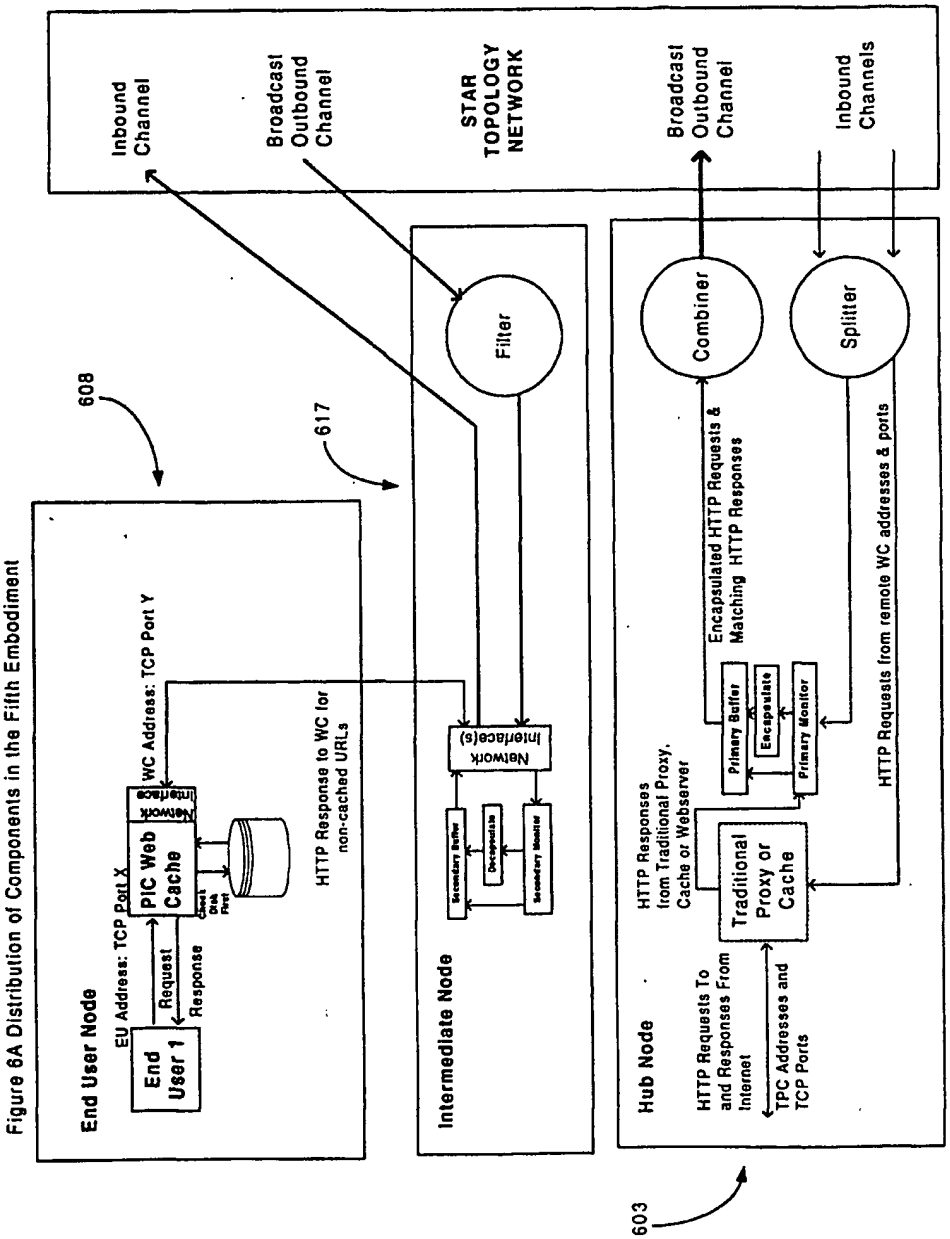


Figure 6A Distribution of Components in the Fifth Embodiment

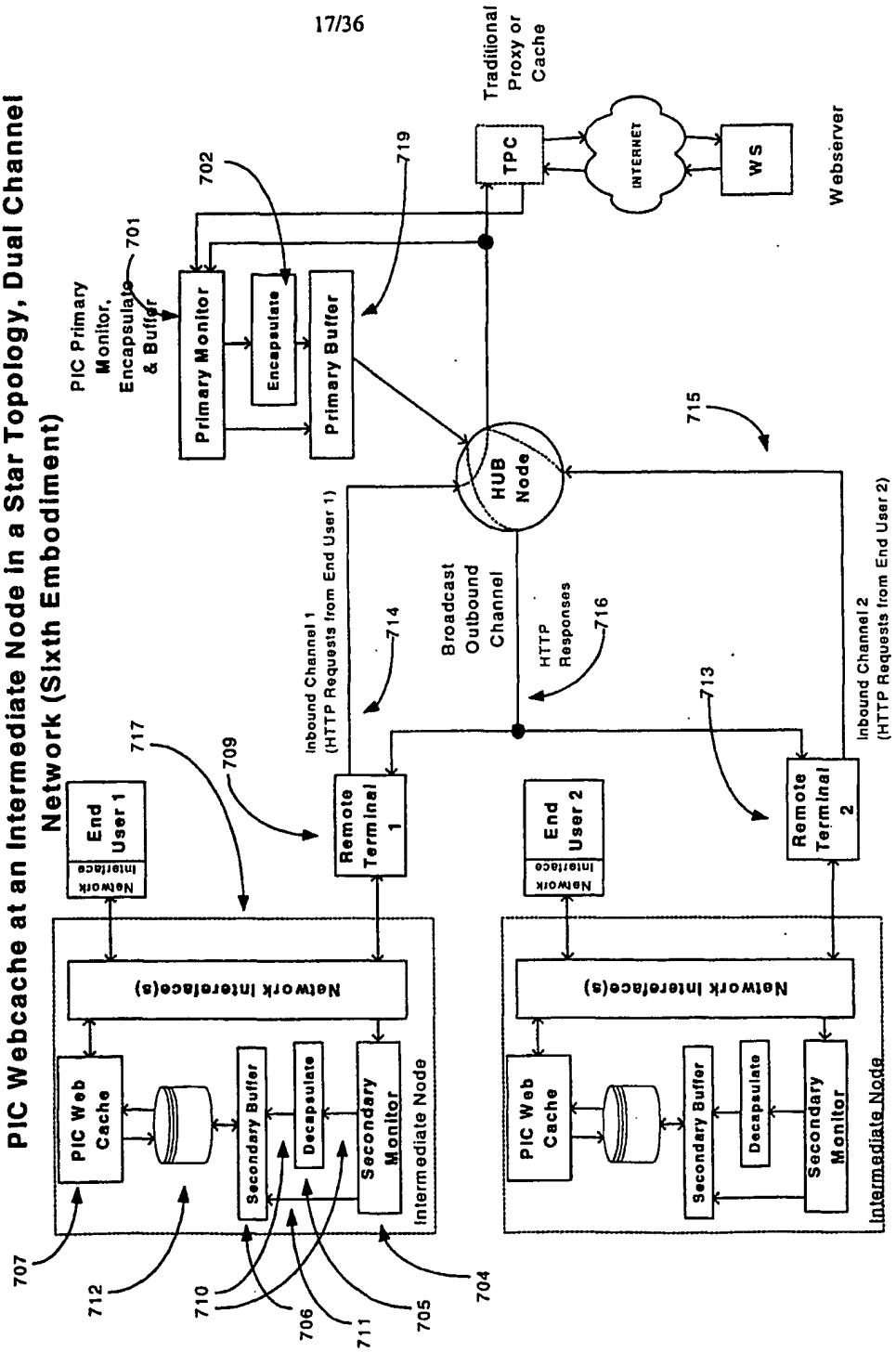


Figure 7. Passive Internet Cache with Socket Matching at the Hub Node and with PIC Webcache at an Intermediate Node in a Star Topology, Dual Channel Network (Sixth Embodiment)

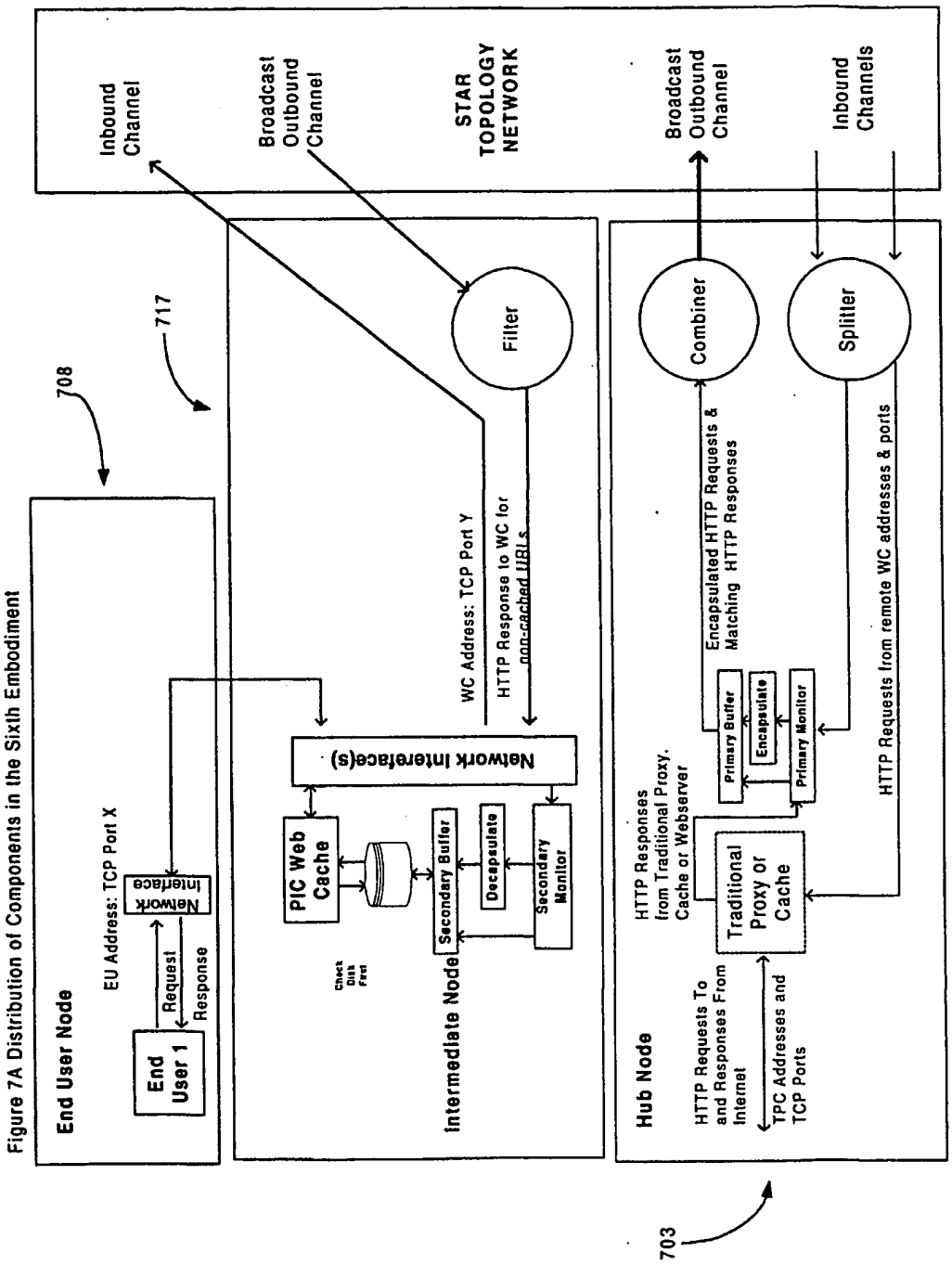


Figure 7A Distribution of Components in the Sixth Embodiment

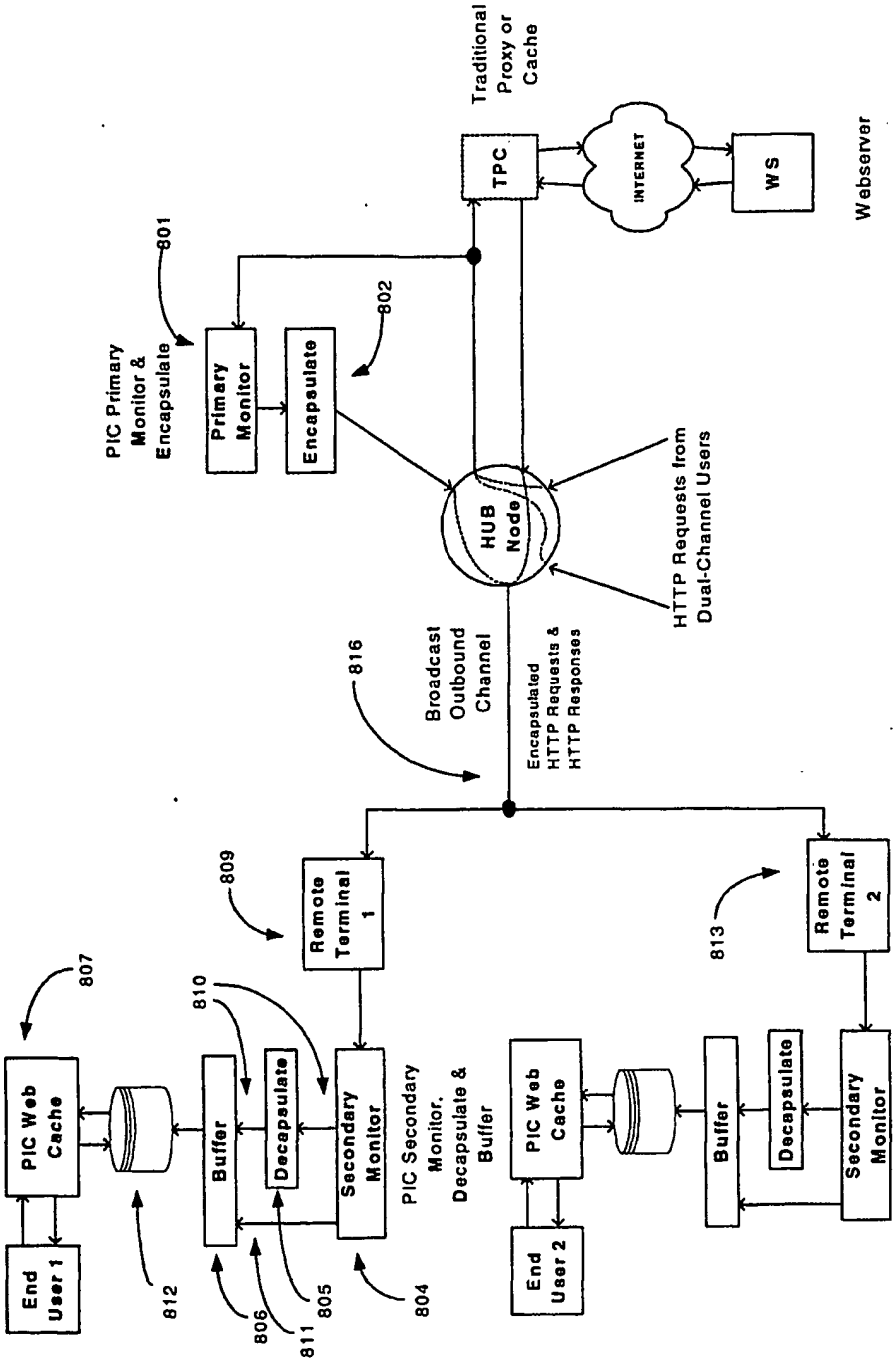


Figure 8. Passive Internet Cache with Socket Matching and PIC Webcache at an End-User Node in a Star Topology, Single Channel Network (Seventh Embodiment)

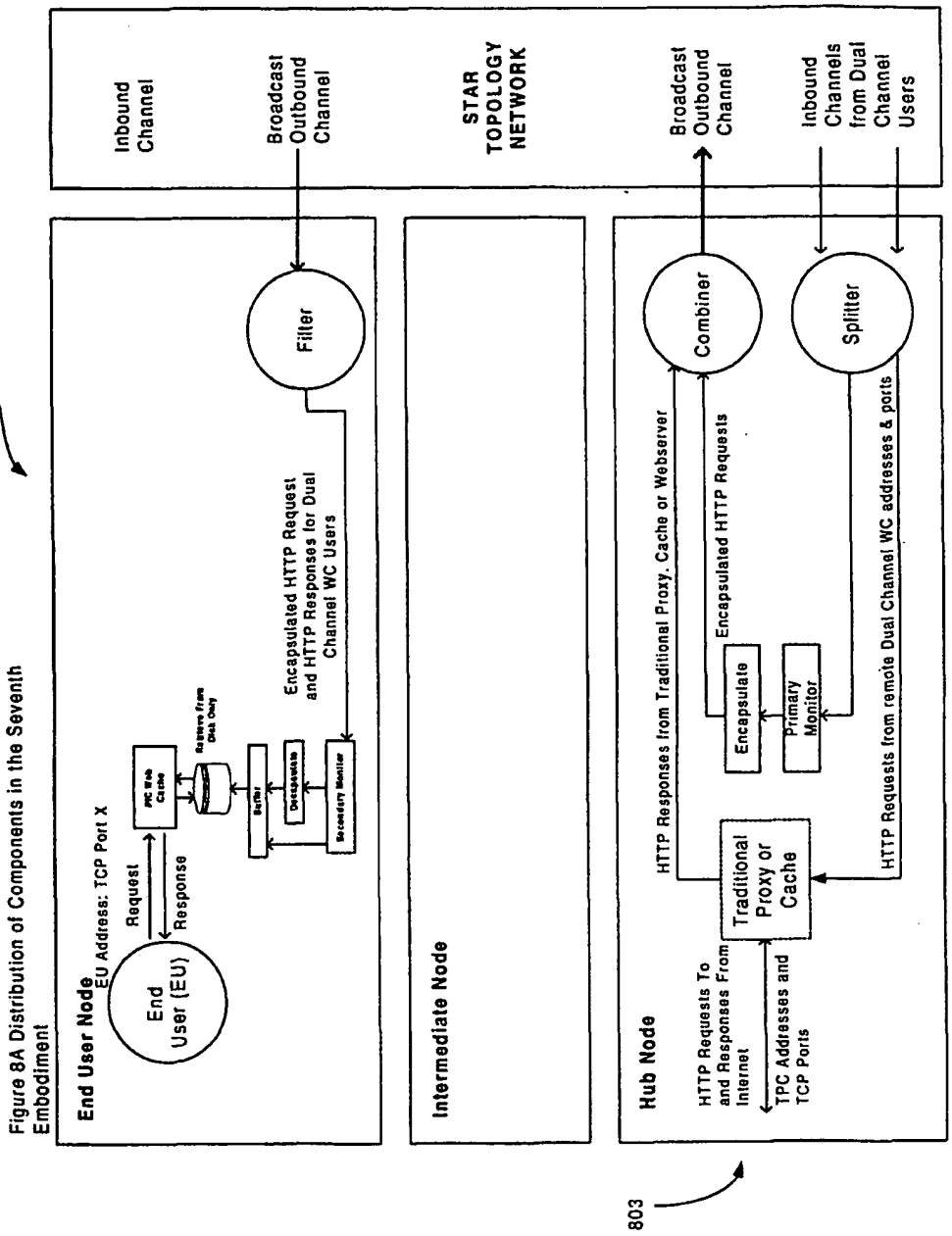


Figure 8A Distribution of Components in the Seventh Embodiment

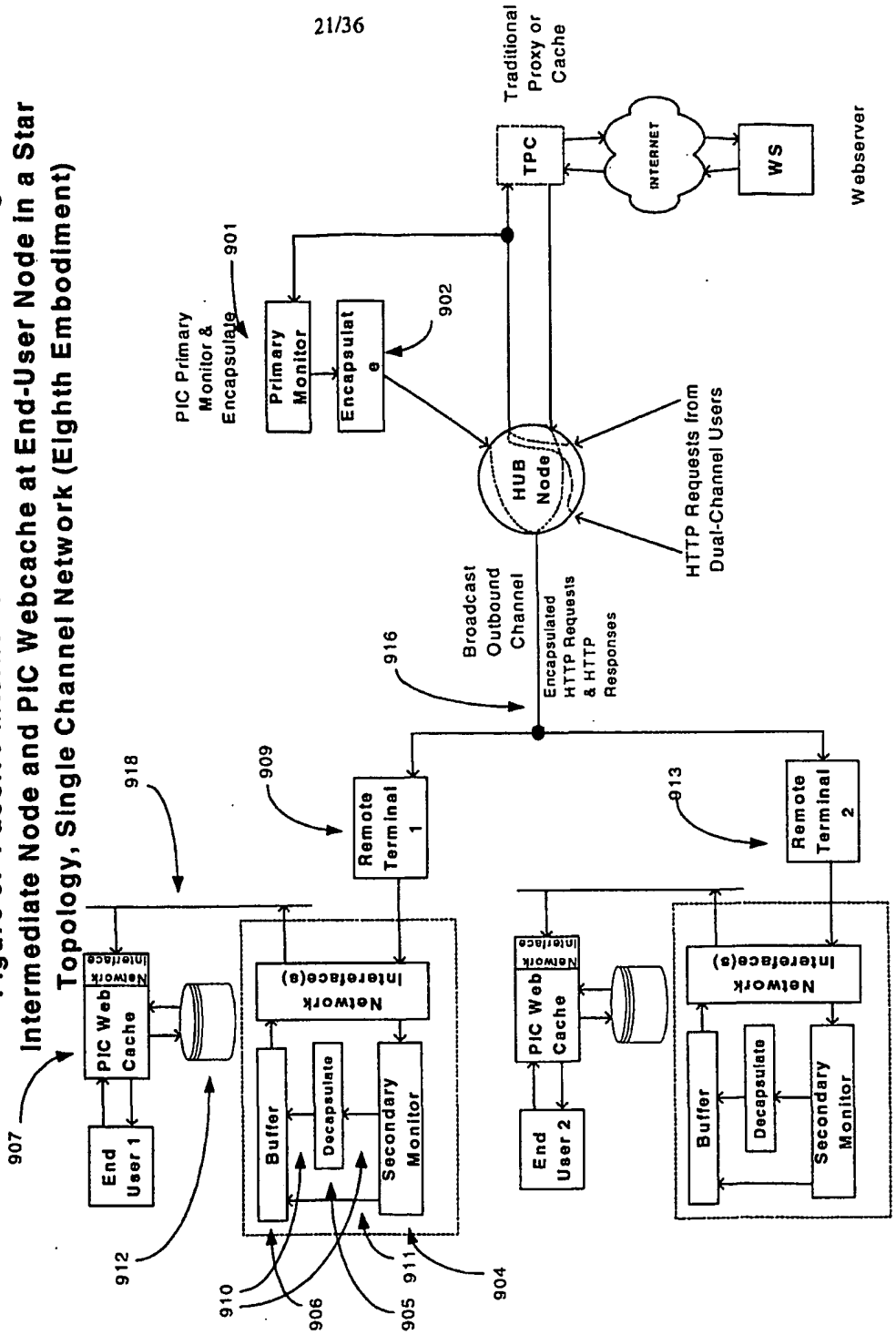


Figure 9. Passive Internet Cache with Socket Matching at Intermediate Node and PIC Webcache at End-User Node in a Star Topology, Single Channel Network (Eighth Embodiment)

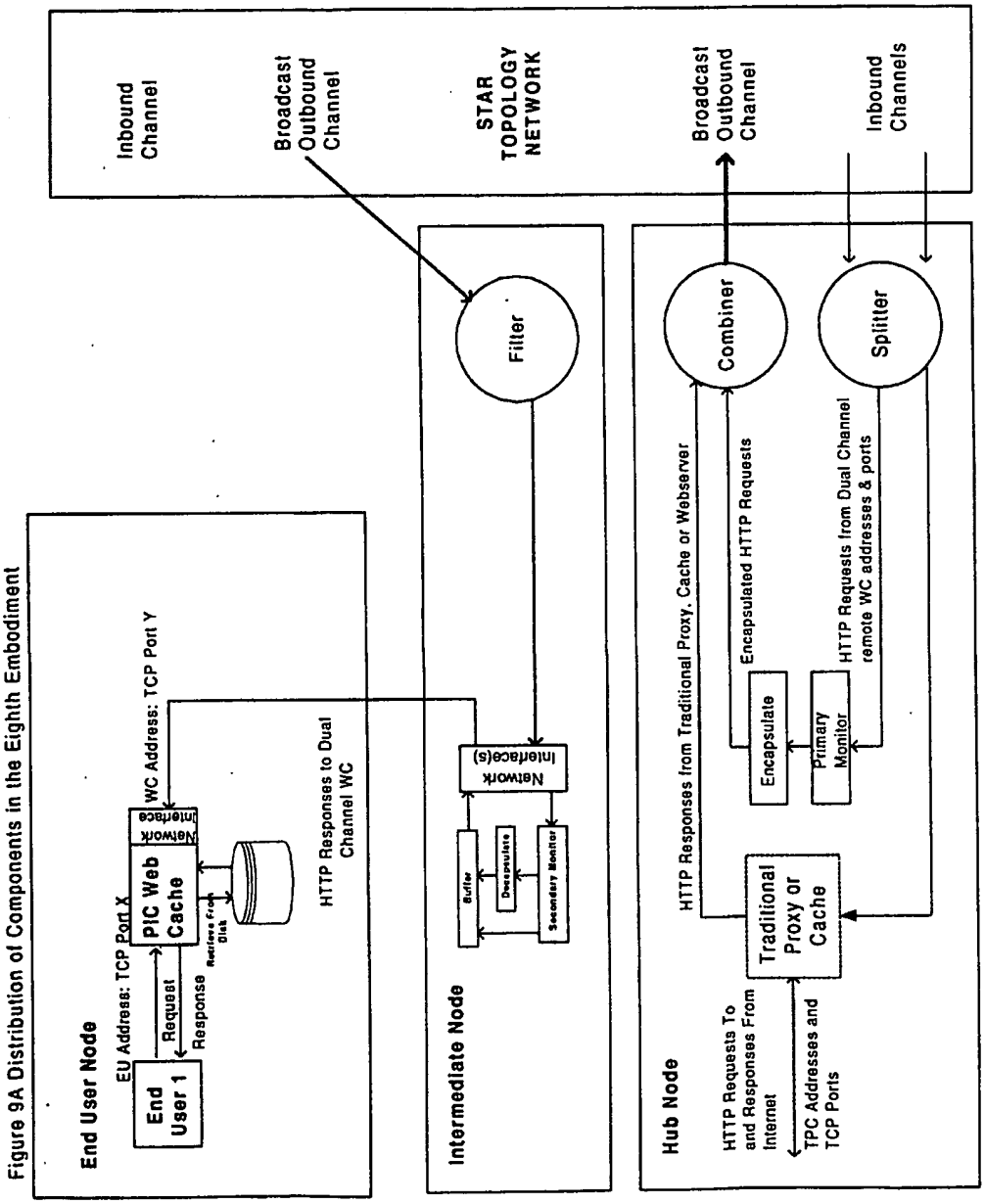


Figure 9A Distribution of Components in the Eighth Embodiment

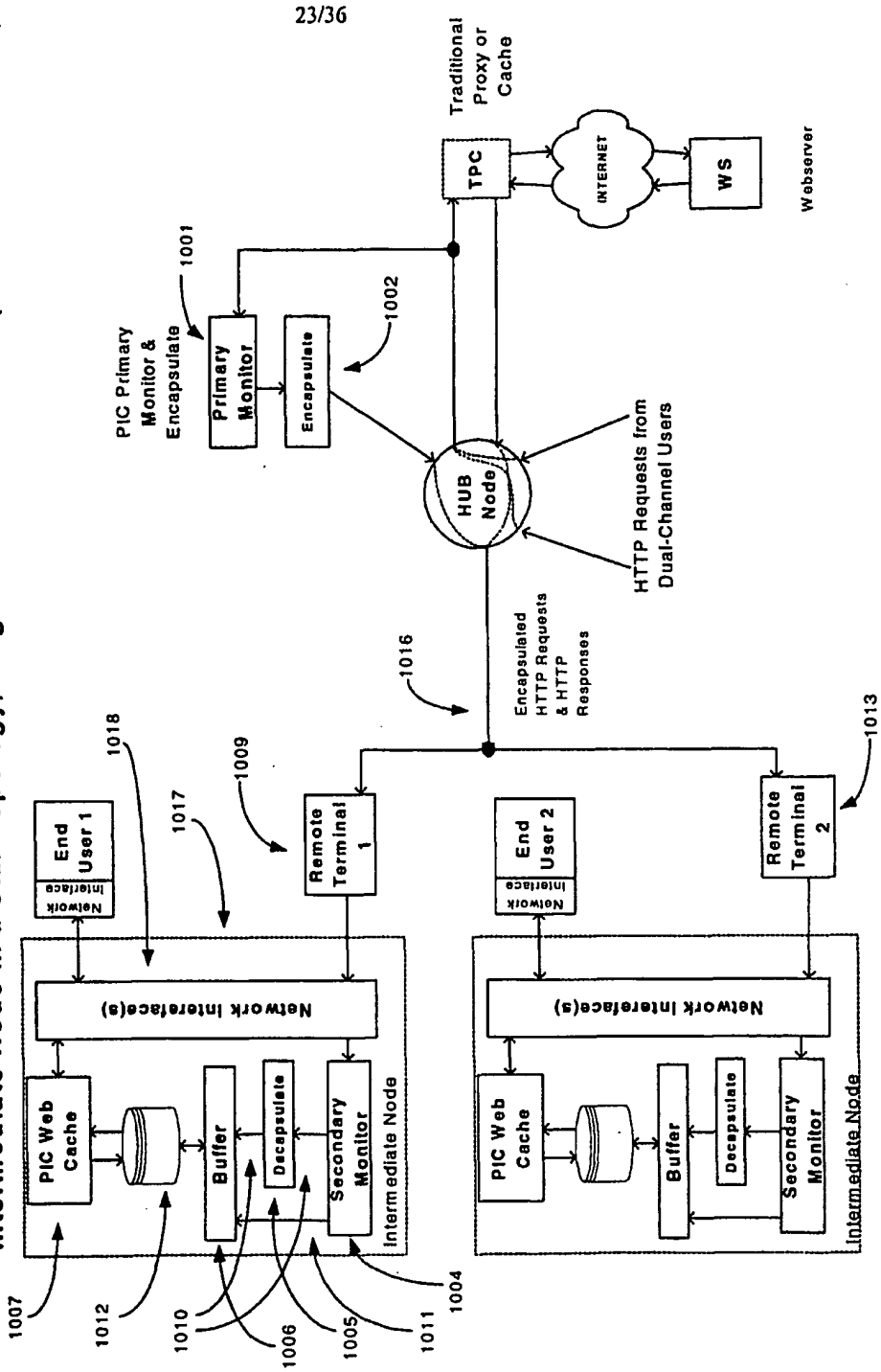


Figure 10. Passive Internet Cache with Socket Matching and PIC Webcache at an Intermediate Node in a Star Topology, Single Channel Network (Ninth Embodiment)

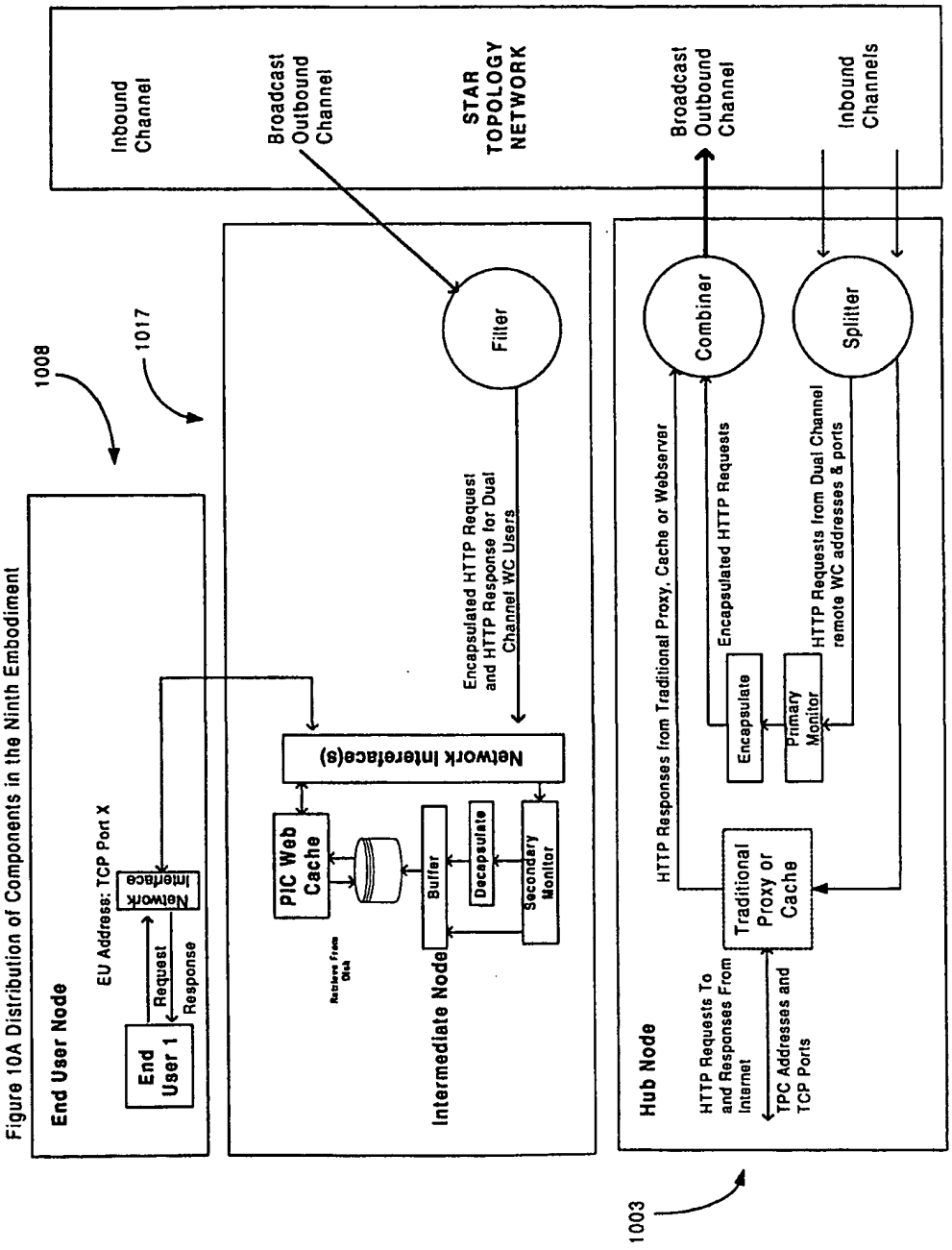


Figure 10A Distribution of Components in the Ninth Embodiment

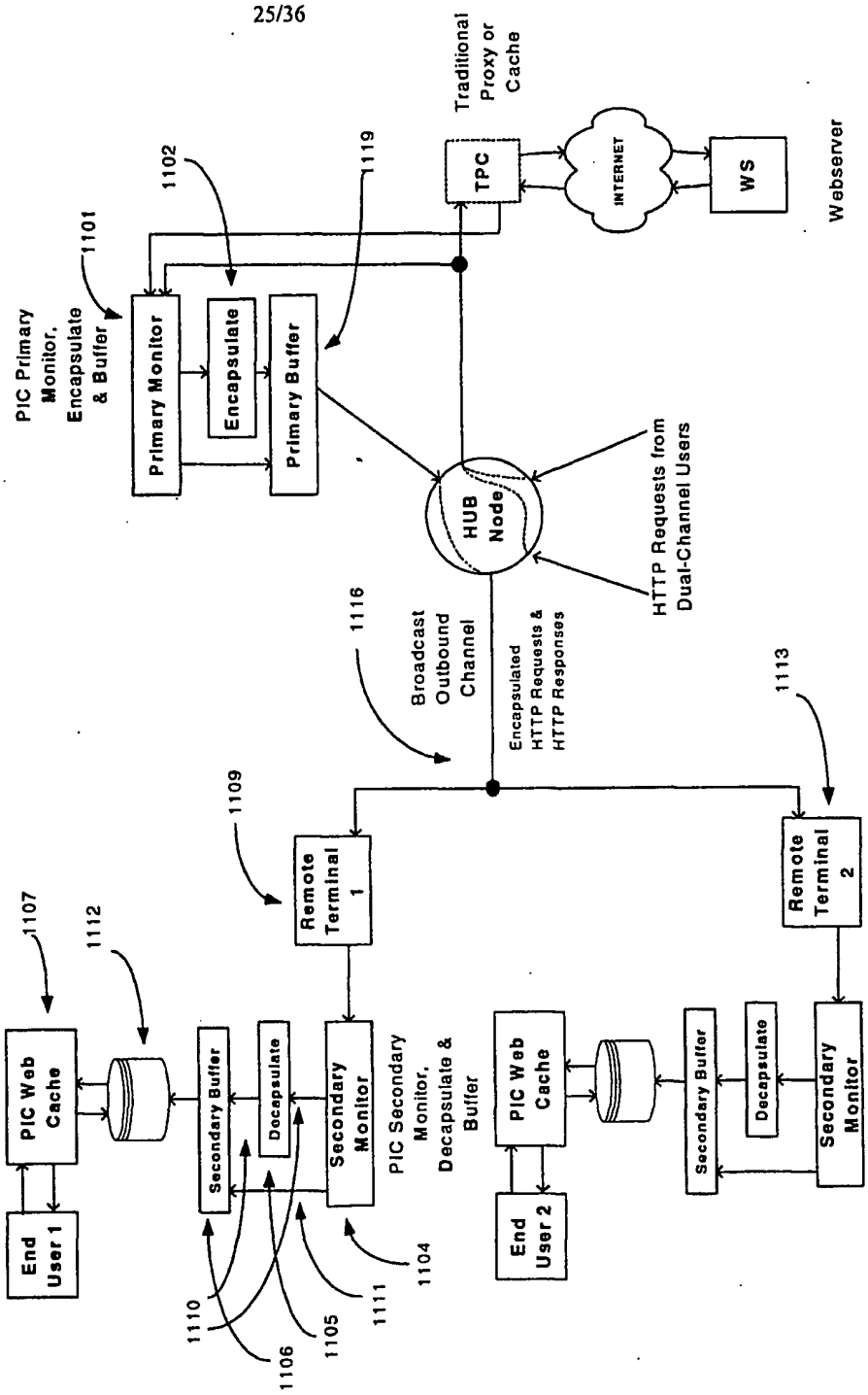


Figure 11. Passive Internet Cache with Socket Matching at a Hub Node and with Secondary Buffering and PIC Webcache at an End-User Node In a Star Topology, Single Channel Network (Tenth Embodiment)

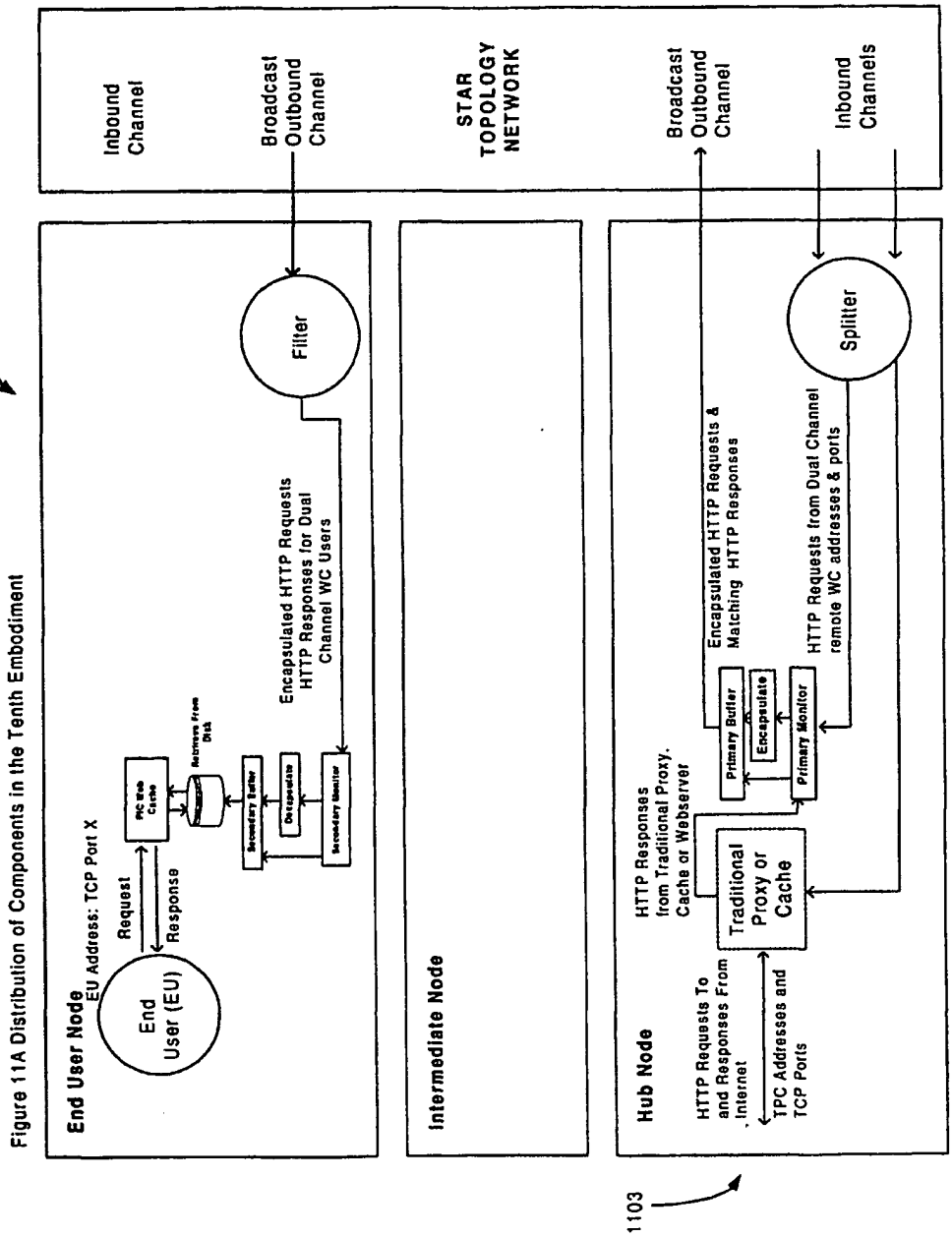


Figure 11A Distribution of Components in the Tenth Embodiment

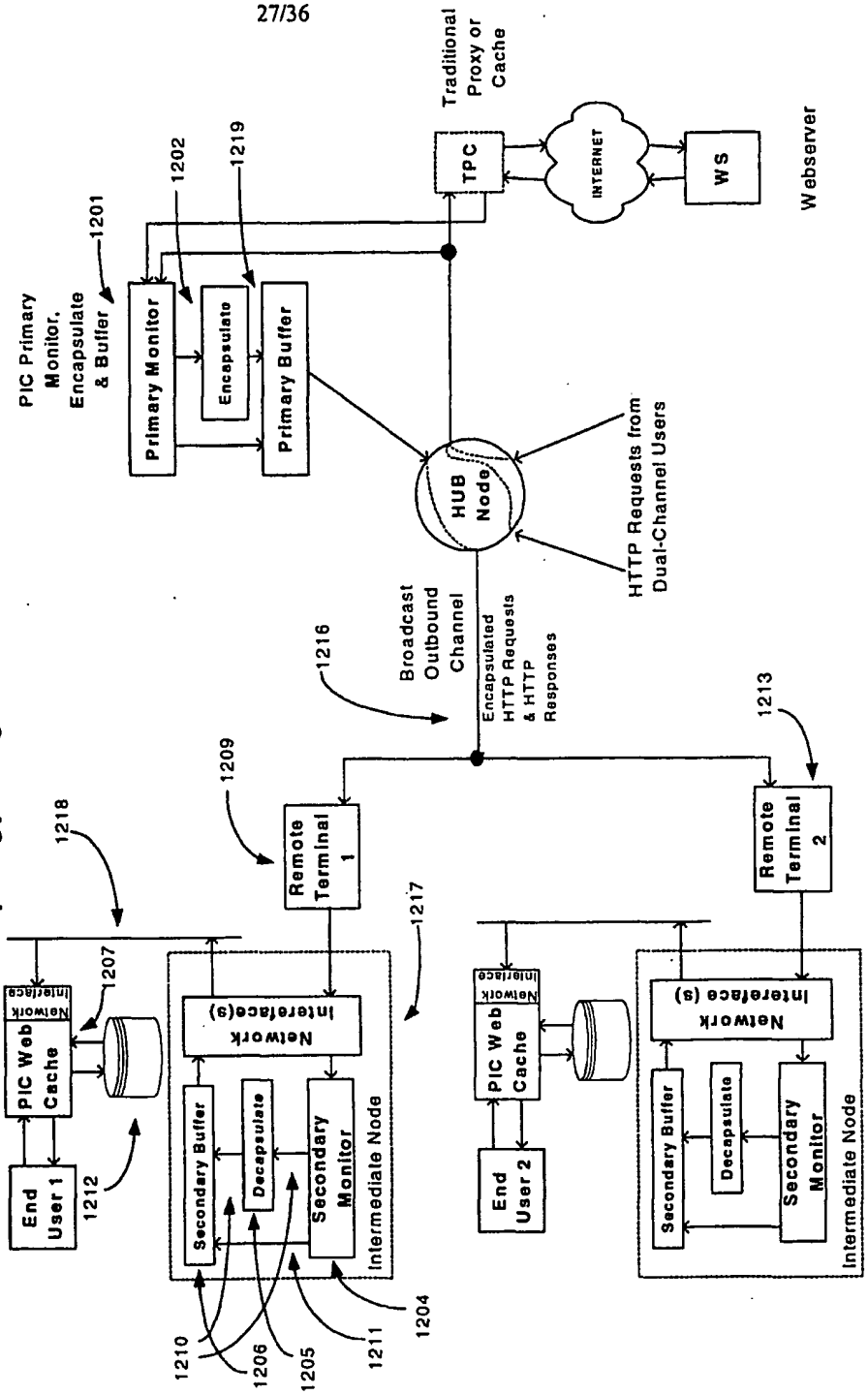


Figure 12. Passive Internet Cache with Socket Matching at the Hub Node, with Secondary Buffering at an Intermediate Node, and with PIC Webcache at End-User Node in a Star Topology, Single Channel Network (Eleventh Embodiment)

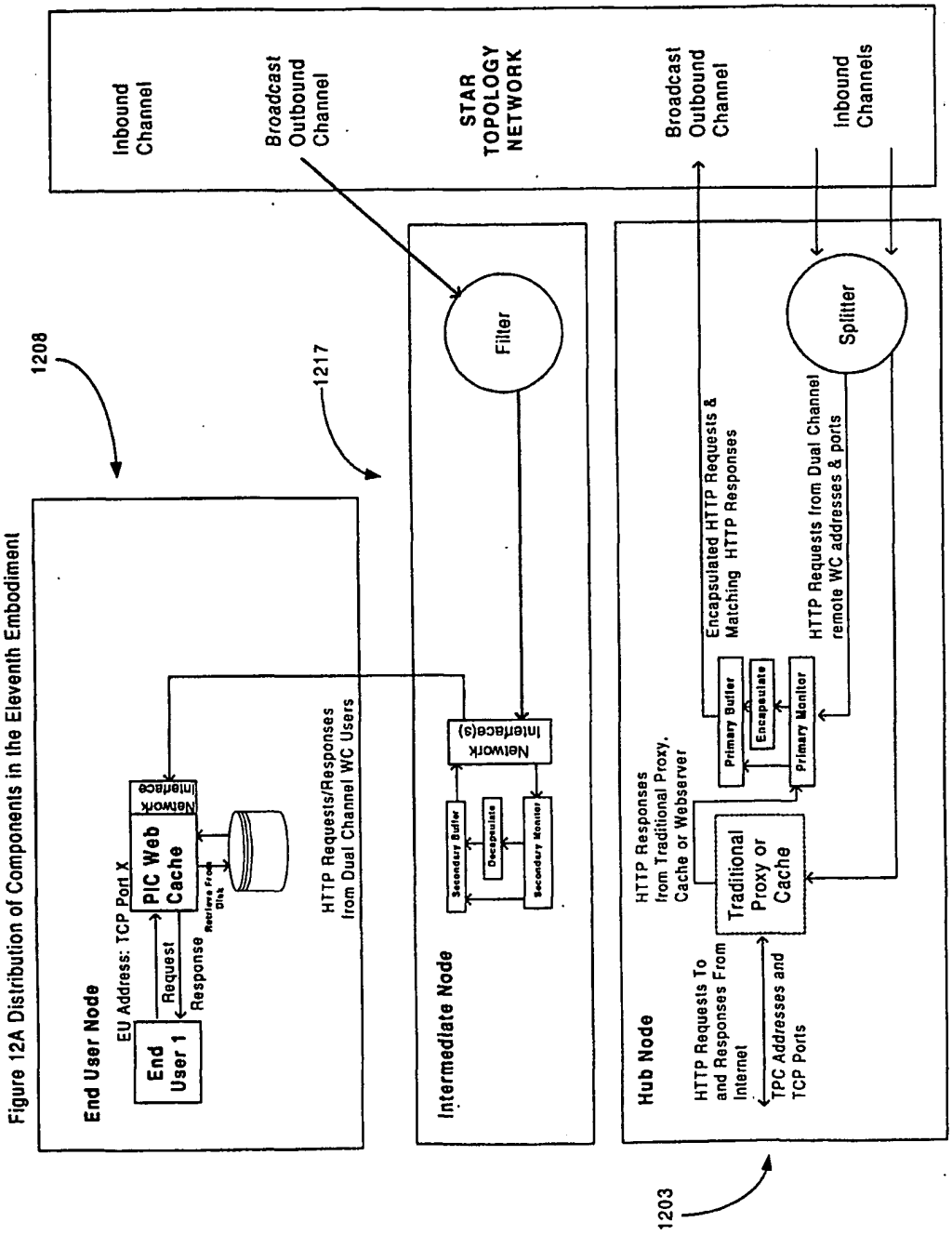


Figure 12A Distribution of Components in the Eleventh Embodiment

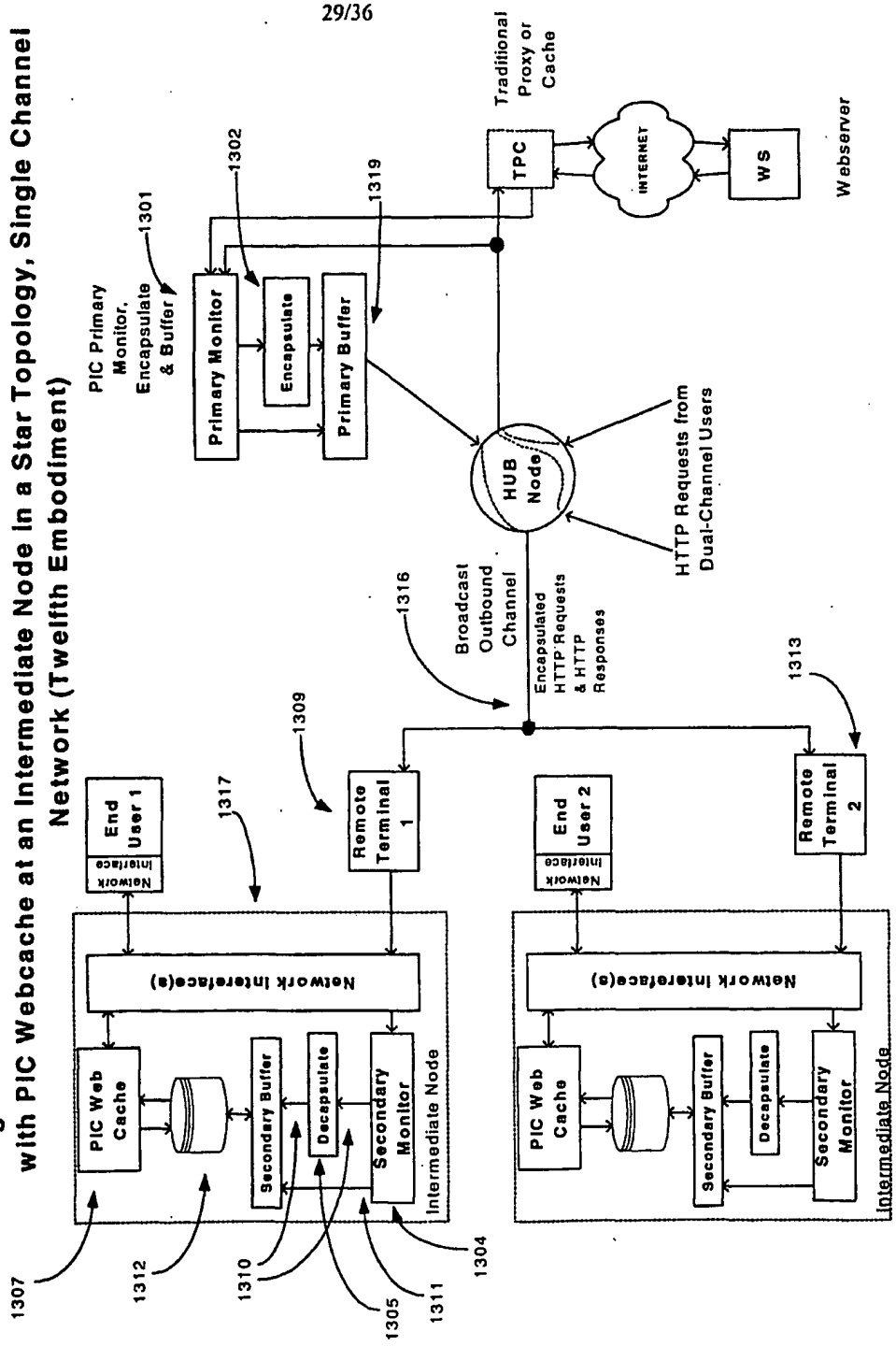


Figure 13. Passive Internet Cache with Socket Matching at the Hub Node and with PIC Webcache at an Intermediate Node in a Star Topology, Single Channel Network (Twelfth Embodiment)

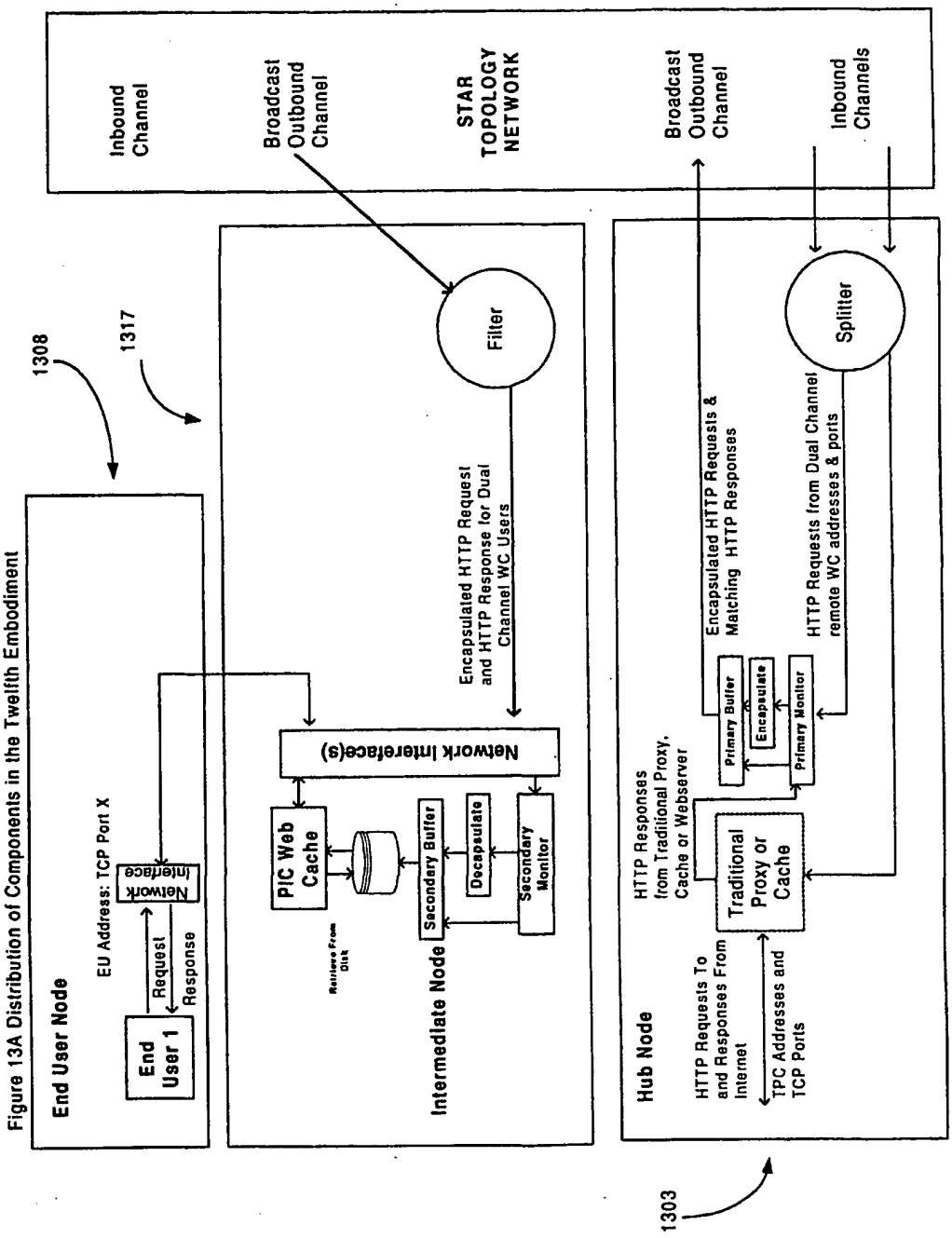


Figure 13A Distribution of Components in the Twelfth Embodiment

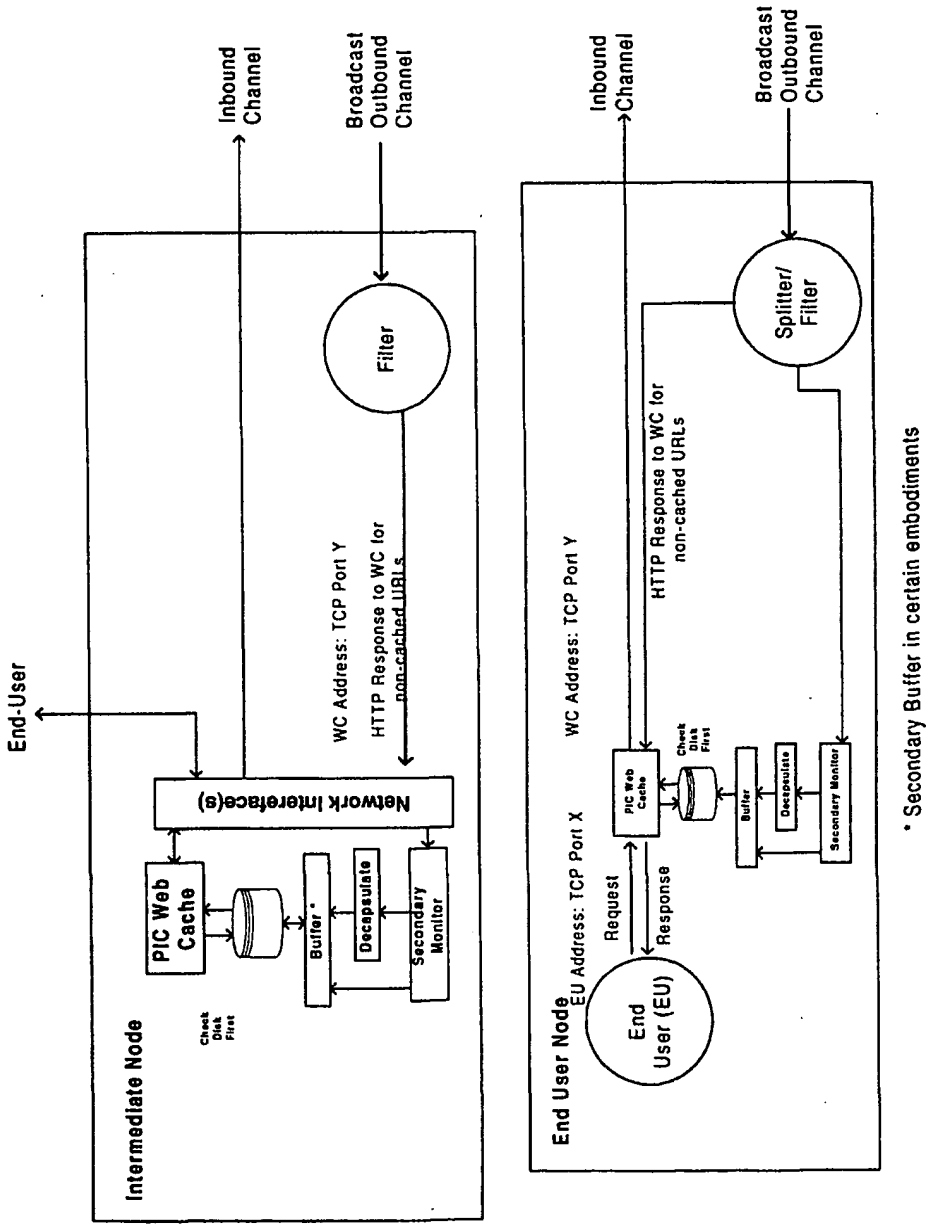


Figure 14 - Block Diagram of Intermediate Node and End-User Node in a Star Topology, Dual Channel Network

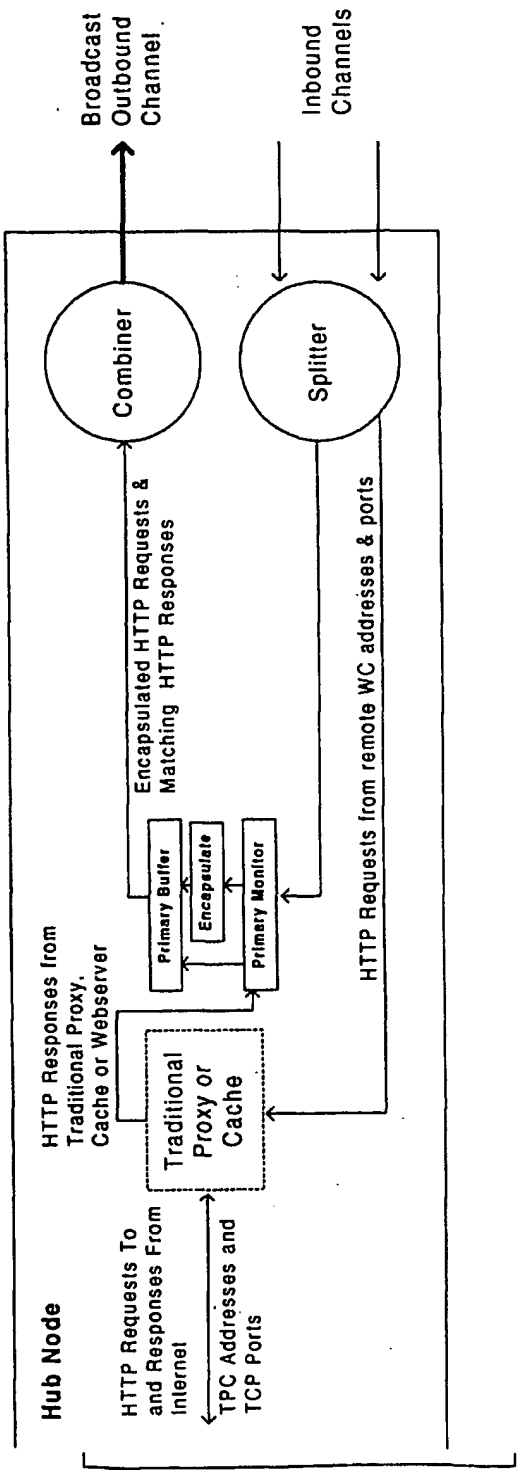


Figure 14A - Block Diagram of Hub Node Functional Modules in a Star Topology, Dual Channel Network with Buffering at Hub Node

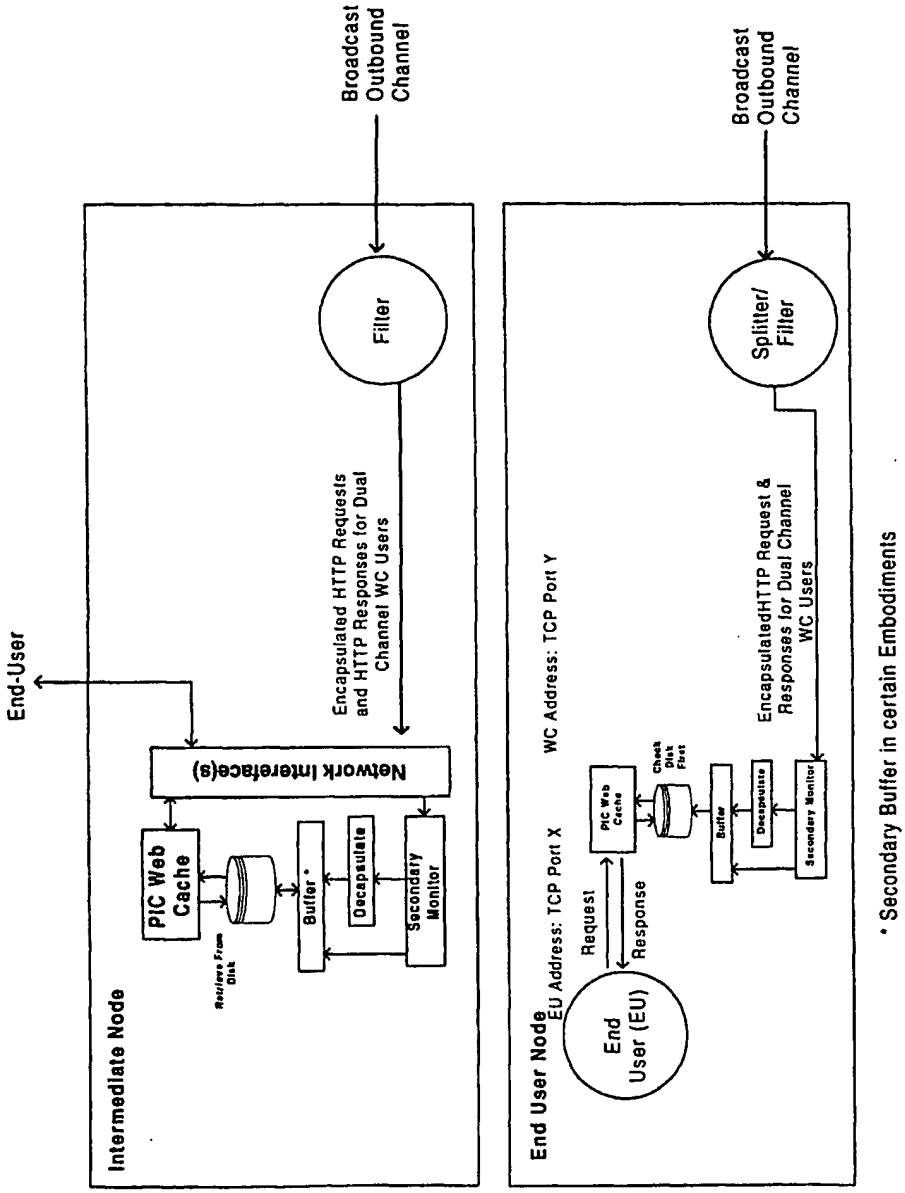


Figure 15 - Block Diagram of Intermediate Node and End-User Node in a Star Topology, Single Channel Network

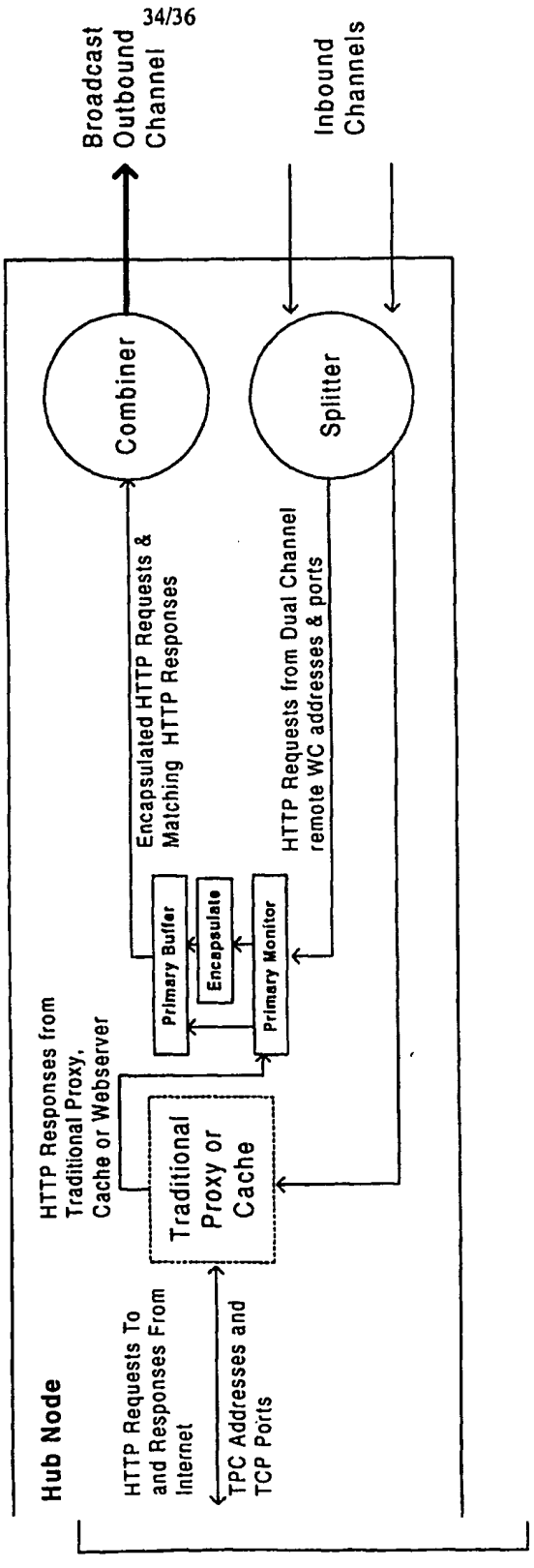


Figure 15A - Block Diagram of Hub Node Functional Modules in a Star Topology, Single Channel Network with Buffering at Hub Node

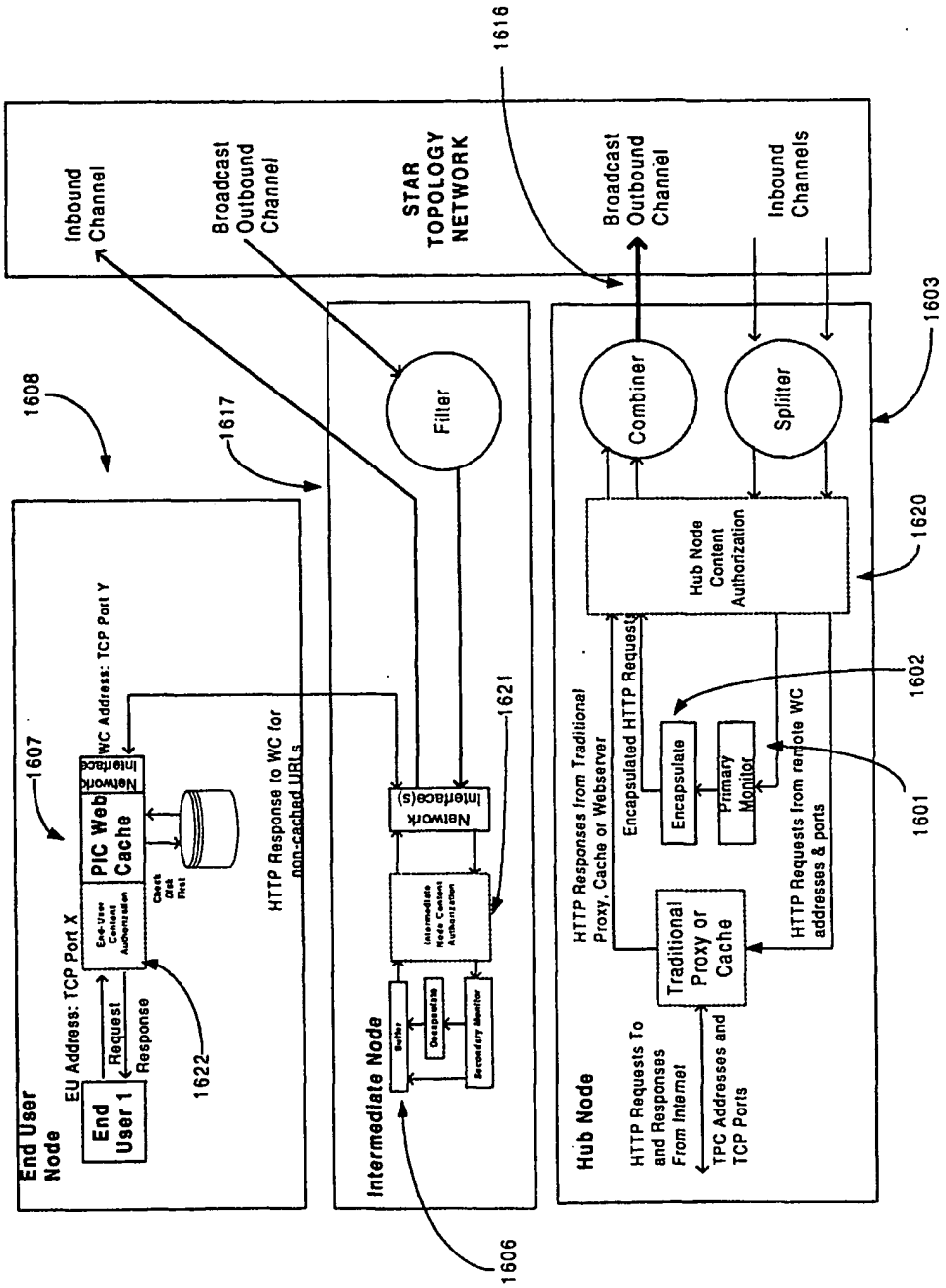


Figure 16 Block Diagram of a PIC System with Content Authorization and with Buffering Component at an Intermediate Node and PIC Webcache at an End-User Node

Figure 17 - Error Correction by Buffering Component Based on Continuity of TCP Sequence Numbers